Computational Modelling of Red Blood Cells

Josef Idris Khan

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

A dissertation presented in part fulfilment of the requirements of the Degree of Master of Science at The University of Glasgow

September 15, 2014

Abstract

A mathematical model has been developed by Lew and Bookchin (1986) to describe behaviour of human red-blood cells under laboratory conditions. These authors also created a computer program to numerically solve the set of equations in their model. This program can simulate experiments in the laboratory and has proven useful over many years. However, the program, written in the BASIC programming language, is now not very portable. Moreover, the program has undergone numerous updates and extensions based on published as well as unpublished work. There is, consequently, a need, in the interest of scientific reproducibility, to update both the computer program and the documentation for the red-blood cell model.

This dissertation reports the results of work carried out to attempt to document the latest version of the revised red-blood cell model, as it is manifested in the computer program, as well as create a new version of the computer program, in a modern programming language.

To carry out the project the original computer program had to be deciphered using the source code, as well as several documents. Unfortunately, this proved to be a difficult and time-consuming task. Consequently, neither the revised documentation nor the new computer program were fully completed. Nonetheless, substantial progress was made. This dissertation reports the results of the research work carried out to update the model documentation and the computer program.

Acknowledgements

I gratefully acknowledge the guidance and advice of my supervisor Dr. Simon Rogers throughout this project.

I thank my client, Professor Virgilio L. "Arieh" Lew, for his patience in answering my many questions on his code and the science behind it. I had hoped that the computer program and documentation produced in this project would prove useful to him, his colleagues, and other researchers in the area of red-blood cell research, but alas, I did not seem to have enough time to complete all parts of the project.

Contents

1 Introduction **5**

2 Background **7**

2.1 Human Red-Blood Cells (RBCs) 7

2.2 Scientific Research on RBCs 7

2.3 The Red-Blood Cell (RBC) Model 8

2.4 The Computer Program of the RBC Model 9

2.5 Project Statement 10

2.6 Rationale for the Project 11

2.7 Literature Resources for the RBC Model 11

2.8 Computer Program Resources for the RBC Model 12

3 Requirements **13**

3.1 Stakeholders 13

3.2 Requirements Capture 13

3.3 Requirements Prioritisation 14

3.3.1 Functional Requirements 14

3.3.2 Non-functional Requirements 16

3.4 Use Cases 16

3.4.1 Use Cases Narrative Description 16

3.5 Analysis of Requirements 17

4 Deciphering the Computer Program for the RBC Model 19

4.1 Programming Language 20

4.2 Program Documentation 20

4.3 Comments in the Program 20

4.4 Variable Labels in the Program 20

4.5 Structure of the Program 21

4.6 Modifications to the Program 22

4.7 Equations used in the Program 23

4.7.1 The Subject of the Equations 23

4.7.2 Different Forms of the Equations 23

4.7.3 Alternative Equations 23

4.8 Algorithms used in the Program, etc., ... 23

4.9 Magic Numbers in the Program 24

4.10 Bugs in the Program 24

4.11 Recalculations in the Program 25

5 The Revised Human Red-Blood Cell (RBC) Model 26

5.1 The Equations used in the Model 26

5.1.1 Equations from Conservation Laws 26

5.1.2 Equations that Define Specific Membrane Transport Mechanisms . . 30

5.2 The Procedure for using the equations in the RBC Model 34

5.3 The Algorithm used to calculate the Reference State of the RBC Model in Lew and Bookchin (1986) 34

5.4 The Algorithm used to calculate the Reference State of the RBC Model in the old program 36

5.5 The Algorithms used to calculate the RS for the new code 37

5.6 The Algorithms used to calculate the Dynamic State for the new code . . . 37

6 Software Design and Implementation **47**

6.1 Software Architectural Pattern 47

6.2 General Considerations 48

6.3 Classes Design 48

6.4 Model Classes/Objects. 50

6.5 View-Controller Classes 50

6.6 The Algorithms used to calculate the Reference State 50

6.7 The Algorithms used to calculate the Dynamic State 51

6.8 Creation of the Code 51

7 Software Evaluation **52**

8 Conclusions **53**

9 Future Work **54**

Chapter 1

Introduction

Science could be described as the systematic study of phenomena to acquire new knowledge based on measurements and logical reasoning. The method used to obtain that knowledge, the scientific method, is a fundamental requirement for the study to be considered science.

The scientific method consists of a series of procedural steps to follow in the process of investigating a subject. These include, for example, carrying out observations or measurements of the subject under investigation, attempting to explain those observations or measurements with a hypothesis or theory, using the hypothesis or theory to make predictions about the subject, carrying out further observations or measurements of the subject to test the hypothesis or theory, analysing and interpreting those observations or measurements, and modifying the hypothesis or theory if necessary.

An important aspect of the scientific method is that another person should be able to independently confirm the results of the investigation to prove their validity. This requires that the entire experiment with observations and measurements be reproducible. That is, another person should be able to carry out the same experiment independently, but under the same conditions, and find the same results (within the margins of experimental error). This concept of scientific reproducibility is a crucial aspect of science and the scientific method. Without it the results of an investigation cannot be considered to have been verified or confirmed.

Science includes a number of areas such as formal sciences, applied sciences, natural sciences, and social sciences. In all of these the scientific method must be followed. The natural sciences include the physical and life sciences, such as, e.g., biophysics and biochemistry. The applied sciences include, e.g., computer science.

The behaviour of a subject under investigation can be described on the basis of the results of a number of observations without a detailed theory, in which case the description would be termed empirical. The alternative approach would be to develop a detailed theoretical description formulated using already established facts and logic, including mathematics.

To be scientifically valid the theory behind the description of the behaviour must be independently testable, or, in other words, falsifiable. That is, a person should be able to independently test whether the theory is valid or not.

A mathematical model of a system consists of a set of equations that describe the system. To investigate the behaviour of the system requires solving those sets of equations. Some mathematical models can be solved analytically, that is by manipulating the symbols in the equations of the model using established rules of mathematics.

However, some problems cannot be solved analytically. Such models could be solved computationally (or numerically). That is, using techniques where approximations are used to solve the equations. A typical approach is to consider a small step in the system, such as a small time-step. During such a step a small change is made to one or more of the variables in the equations and the changes in other variables are calculated. This is repeated for many steps to determine the behaviour and development of the system. Computational models, generally speaking, require the use of computers to solve the model.

Computational models are important because sometimes the system is too difficult to solve analytically. This may be because the system is too complex to be solved analytically, or an analytical solution is possible but too time-consuming to carry out. An example is a model which involves numerous equations with a large number of variables. Computational models are useful because they can be used to solve a wide range of problems. Furthermore, the simulations made by running the computational model can be considered to be computational experiments. These can often be compared directly with actual laboratory experiments. Consequently, some computational models can be used to predict behaviour or results in experimental studies.

The project discussed in this dissertation is concerned with a computational model for red-blood cells. The mathematical model has been modified numerous times, and, unfortunately, the complete model is not fully documented anywhere. Consequently, one aim of this project is to document that updated and modified red-blood cell model. In addition, while a computer program for this computational model does exist, it has been created using an older computer programming language, which has a number of disadvantages. Thus, another aim of this project is to re-write the old computer program as a new computer program which uses a more modern programming language.

In Chapter 2 a brief description is given of the red-blood cell (RBC) model, including published and unpublished work on that model. Furthermore, a brief account of the current computer program is also given. The purpose and rationale for the project are outlined. Requirements engineering aspects of the project are discussed in Chapter 3. Chapter 4 describes the main difficulty in this project, namely, the problems encountered deciphering the old computer program. Using the various resources available and discussed later the updated complete RBC model is fully documented. Chapter 5 presents this documentation. The design and implementation of the software product produced in this project is described in Chapter 6. An evaluation of that product is provided in Chapter 7. The main conclusions of the project are given in Chapter 8, while possible future work is described in Chapter 9.

Chapter 2

Background

2.1 Human Red-Blood Cells (RBCs)

Human red-blood cells have a relatively simple structure compared to most other human cells. The outer surface of the cell is the cell membrane and within this lies the gel-like cytoplasm. In human red-blood cells the cytoplasm consists predominantly of the protein haemoglobin which transports oxygen from the lungs to the rest of the body as the cells flow as part of the blood pumped around the circulatory system. The haemoglobin in red-blood cells also removes carbon dioxide from the body transporting it to the lungs to be expelled. Mature human red-blood cells do not contain a nucleus, nor do they contain inclusions in the cytoplasm that occur in many other cells. As with other cells, the red-blood cell membrane facilitates the binding and transportation of ions and molecules between the cell cytoplasm and the extra-cellular fluid.

The properties of a red-blood cell, such as the concentration of substances within it, can change in response to changes in the environments inside and outside of the cell. Typically, studies of red-blood cells examine the properties of the cells due to changes in the intra-cellular and extracellular environment, as well as due to the various transport and binding mechanism on the cell membrane.

2.2 Scientific Research on RBCs

Scientific research into human red-blood cells in the laboratory is an important area of research that is relevant to the study of human physiology and to the study of diseases of the blood, such as, e.g., sickle-cell anaemia.

Lew and Bookchin (1986) developed a mathematical model to describe the processes involved in human red-blood cells. This red-blood cell (RBC) model is particularly useful for comparisons with laboratory studies of such cells. As part of the modelling these authors also created a computer program to numerically solve the equations in their mathematical model. The paper by Lew and Bookchin (1986) describes both the mathematical model and the results produced using the computer program.

While there are many areas of research in RBCs, this project is concerned specifically with the model of Lew and Bookchin (1986) and subsequent developments of that model.

2.3 The Red-Blood Cell (RBC) Model

The mathematical model of Lew and Bookchin (1986) was the first model designed to incorporate, as far as possible, all aspects relevant to the study of red-blood cells into a single model. Previous models of red blood cells had only examined a limited number of aspects. The mathematical model of Lew and Bookchin (1986) was designed to model mature red blood cells, which are also known as erythrocytes. In particular, its purpose was to incorporate, as far as possible, all aspects relevant to the study of such cells into a single model. Previous models by other authors had only examined a limited number of aspects. The integrated model of Lew and Bookchin (1986) was an important breakthrough enabling the changes to the properties of red-blood cells to be studied comprehensively and in detail for the first time.

In order to make the mathematical model of Lew and Bookchin (1986) tractable it was designed to be comparatively simple. The model has now been used extensively since its creation to study the behaviour of mature red-blood, which are also known as erythrocytes.

The model of Lew and Bookchin (1986) was later modified by Lew et al. (1991) to study other blood-cells, namely immature red-blood cells, known as reticulocytes, which typically survive for about 1 day in the blood, compared to about 120 days for mature red blood cells (erythrocytes). The model of Lew et al. (1991) is relevant for the study of sickle cell anemia.

Since its creation the computer program for the RBC model has been modified many times, over many years. These revisions to the program were made to take into account further refinements in the RBC model. Some of the changes or developments of the RBC model have been published, e.g., Lew et al. (1991), which, for example, included, among other things, transport processes across the cell membrane, which were not included in the original model.

In addition, other changes to the program have been made based on as yet unpublished work, such as an extension of the RBC model to take into account of the effects of magnesium and calcium transport across the cell membrane.

Furthermore, some of the changes to the computer program have involved not only the inclusion of additional effects, but also modifications of how some of the original effects were implemented. Consequently, at present, the latest revised version of the RBC model, as represented by the current computer program of that model, is not fully documented anywhere.

As mentioned in the Introduction, it is important that scientific results are reproducible. Thus, to confirm their validity, the results of scientific research should be open to independent scrutiny and testing. This applies not only to experimental work, but to theoretical work too.

Consequently, it is important from the point of scientific scrutiny and reproducibility that the latest revised version of the RBC model be documented.

2.4 The Computer Program of the RBC Model

As mentioned in the previous section, the latest version of the RBC model is not documented in a published or an unpublished article. Although the complete RBC model is not fully documented, another possible approach to fulfill the need for full scientific scrutiny of the model might be via the computer program for the RBC model. Specifically, it might be possible to determine the model either from separate documentation for the computer program, from comments within the program itself, or following the code regardless of comments.

Although the computer program has undergone numerous modifications, the documentation for the computer program has not been kept up-to-date. Thus, the model cannot reliably be determined from such documentation.

While it is true that the computer program does contain the latest version of the RBC model there are a number of difficulties with the program which make it difficult for a researcher to ascertain the model from it. The problems mainly relate to the following points: the particular programming language chosen and how the computer program was implemented. These are briefly described below.

The computer program is written in an older programming language, namely, the BASIC (Beginner's All-purpose Symbolic Instruction Code) programming language. The particular dialect of BASIC used to create the computer program is the proprietary PowerBASIC, which is based on the earlier TurboBASIC. The BASIC programming language in which the computer program was constructed is now not very popular nor readily available for a range of computers. Other programming languages have become more popular. Most potential users of the computer program for the revised RBC model, other than Prof. Lew, will, likely, not be familiar with the BASIC programming language.

However, even if a user is familiar with, or learns BASIC there still remain a number of impediments to overcome with the current computer program. Each of the problems mentioned below means that fully understanding and modifying the code is not easy for such a user.

Generally, speaking, the computer program is not particularly well-commented. While parts of the code are reasonably well-commented, other parts are not. This means that a user of the current program has to attempt to decipher that code, which is time-consuming and comparatively difficult.

The structure of the computer program is relatively difficult to follow. The code does not use modern object-oriented design techniques, that is, it does not have a modular design, which would make it much easier to follow and modify. Although PowerBASIC does, apparently, support object-oriented programming, the old code used here was not written using object-oriented programming. This is undoubtedly because the code was

written before object-oriented programming was popular. Later changes to the code did not modify this overall design and approach.

As already mentioned the computer program has been modified incrementally over a number of years. Many of the changes have been added in a somewhat unsystematic manner. Consequently, the computer program does not have a particularly clear, or easy-to-follow structure.

Ideally, a researcher in the field should be able to independently check and verify the validity of the model in the computer program. As well as following the logic of the model in the code such a user should be able to use the code. However, a significant problem with the computer program is that it is not portable, that is, it is not usable on many different types of computers. The compiled version of the computer program can be run without an emulator on older 32-bit computers running Windows. However, the compiled executable file can be also be run on more modern 64-bit machines, running Window, MacOS or Linux using an emulator such as DOSBox (http://www.dosbox.com). The raw BASIC source code itself could, possibly, also be run on more modern Windows machines using a more modern PowerBASIC compiler. However this has not been checked. It requires the purchase of the proprietary PowerBASIC software. Whether the raw BASIC source code could be run with other BASIC compilers on more modern computers is unclear to this author. This has not been investigated. Typically, current users of the computer program run it on older 32-bit Windows machines. Such machines are becoming less common. This means that there are difficulties running the computer program unless an older computer is available.

Moreover, modifications to the existing code are difficult due to the choice of the programming language, the lack of full comments in the code, and the overall structure of the code, which have already been mentioned in the context of following the model in the code.

The above observations indicate that the computer program could benefit by being written in a more modern, portable and freely available programming language.

2.5 Project Statement

Basic considerations of the current state of the documentation for the RBC model and the current state of the computer program of that model, which are both mentioned above, indicate there is a clear need for an improvement for both.

Consequently, the aim of this project is two-fold:

- to document the latest version of the RBC model
- to re-write the computer program for the revised RBC model using a modern programming language

Detailed requirements for this project will be discussed in Chapter 3.

2.6 Rationale for the Project

The revised RBC model and its computer program remain very useful to researchers in the field. The manifestation of the model in code form enables a wide range of numerical simulations to be carried out that are relevant to many experimental studies of human red-blood cells.

It is therefore important, in the interests of open scientific scrutiny, that the revised RBC model and its computer program should be available to and used by the widest possible audience of researchers. To enable this, the complete revised model, as it now stands, needs to be fully documented.

In addition, to overcome some of the drawbacks in the current BASIC computer program, the code needs to be re-written. Specifically, the program needs to be written using a more modern programming language, using modern programming techniques, with informative program commenting and clear program documentation.

To create the new program for the revised RBC model, along with documentation for the model and the program, requires that the full and relevant details of the model itself be documented. To do this the literature on the RBC model, both published and unpublished was examined. Furthermore, since it is the current BASIC computer program that contains the most up-to-date version of the RBC model, it was necessary to thoroughly analyze that code in combination with an examination of the documentation. However, for reasons that will be explained in Chapter 4 a major difficulty in this project had been to decipher the BASIC computer program.

2.7 Literature Resources for the RBC Model

In order to determine the details of the current version of the RBC model (as manifested in the BASIC computer program) several papers were examined which describe different aspects of the model. These are described below.

The original RBC model of Lew and Bookchin (1986) has already been mentioned briefly. This article was the most important piece of literature in this project. The equations and procedure presented in Lew and Bookchin (1986) were helpful in identifying some of variables and parameters in the BASIC computer program as well as the procedures used to solve the equations. Comparisons with the computer program, however, revealed some differences in the procedures and the equations used.

The reticulocyte model of Lew et al. (1991) was also helpful in clarifying some of the terms used in equations in the computer program. Although the RBC model does not attempt to model reticulocytes, per se, some of the equations used in that model were helpful in discerning terms and equations in the code. For example, that model was helpful in identifying the addition of transport effects due to Na:Cl and K:Cl cotransporters.

Furthermore, the unpublished draft manuscript by Weiss and Lew (2014) on a cellular homeostasis simulator also provided a useful description of the variables, parameters, pro-

cesses and equations involved in the RBC, from a more general perspective.

Examination of the computer program also indicated new material was added to include calcium and magnesium in the model. An an unpublished draft manuscript by Lew (2014a) on the calcium and magnesium extension to the model was helpful in elucidating those aspects of the code.

Furthermore, electronic communication with the client, Prof. Lew, helped provide clarification on a number of points in the papers and in the computer program. This communication, moreover, led to agreement on some minor changes for the proposed new code.

2.8 Computer Program Resources for the RBC Model

There was some documentation pertaining directly to the code that was useful.

The main resources used to analyse the code were the BASIC source code itself, called M1.BAS, see Lew (2014b). This was created by Lew and Bookchin (1986) and modified by Prof. Lew over many years (from 1986 until the present). This source code can be opened using a text editor and examined directly.

Furthermore, Prof. Lew provided two unpublished pieces of printed information that were helpful in understanding the computer program. One was a set of notes on modelling cellular homeostasis by Weiss and Lew (2014) which was mentioned earlier, while the other was a general guide to using the BASIC program in practice by Lew (2000).

In addition, a precompiled executable version of the program was provided by the client, called M1.EXE To run the computer program for the RBC model this executable file was run on an old laptop running the Window XP operating system.

Chapter 3

Requirements

3.1 Stakeholders

There are two types of intended stakeholders for the software (computer program) and documentation products to be developed. Both types of stakeholders are envisaged to be researchers in the field of RBCs who are familiar with the science in that field.

One type of stakeholder will be assumed to unfamiliar with the Java programming language. Such stakeholders will be referred to as users. They will be able to use the software, follow the documentation for the model, but not necessarily follow or modify the Java code itself with confidence. The client for this project, Prof. Lew, is considered to be a user, and is the main stakeholder.

Another type of stakeholder will be researchers who are familiar with the Java programming language. Such users will be referred to as programmers. As well as being able to use the software and follow the documentation for the model, these users will be able to examine the code directly and modify it. They will be able to the validity of the model in the code and change it.

3.2 Requirements Capture

The principal way that the requirements for the system were determined was by (electronic) communication with the client, Prof. Lew. The basic intention is that the new computer program must, as far as possible, provide all of the functionality of the original computer program. However, for this program to be created and for it to be used effectively also requires that the latest revised version of the RBC model be fully documented.

3.3 Requirements Prioritisation

3.3.1 Functional Requirements

After the requirements for the system were identified they were prioritised. The MoSCoW prioritisation approach, commonly used in software development, was used to specify the importance placed on each requirement. That is, the requirements are identified as the following:

Must: those that are essential and must be implemented for the system to work properly

Should: those that are desirable, but not essential for the product to work properly

Could: those that could be implemented, if time permits

Won't: those that are possible, in principle, but won't be implemented.

In the description given below the functional requirements along with their prioritisation. In addition a brief summary and rationale for each requirement is also noted with bullet points.

The functional requirements listed below have been identified for the system. The system/product:

1. Must:
   (a) carry out the same calculations as the original code (except where agreed with the client)
      - necessary so that the new code can be use as a replacement for the old code

   (b) allow the user to input and modify the variables and parameters in the model that are modifiable in the original code (via a GUI interface)
      - necessary so that the new code can be use as a replacement for the old code

   (c) give the same numbers in the output data as the original code for the same experimental conditions
      - to ensure that the code is working correctly

   (d) output the data in a similar format and to files as is done in the original code
      - to enable the results of the new and old codes to be compared, to confirm the new code gives correct results, and to facilitate users following their previous approach of importing the results into Microsoft Excel to plot out the results. (In the future the output formats could be changed).

   (e) have the complete RBC model documented
      - so that the revised model can be examined and checked by others

   (f) be written using a modern programming language that is platform-independent
      - to ensure that the software is portable, i.e., usable on a wide range of computers with different operating systems.

(g) be written using a modern programming language that is freely available

- to support the principle that the software available to the widest possible audience

(h) be written using object-oriented programming techniques

- to provide a logical, easy to follow structure, with a modular design, so that future modifications readily be carried out by programmers

(i) have the program written with detailed comments in the code

- to enable the program to be easily understood and modified by programmers

2. Should:

(a) include, if possible, all options for modifying the experimental conditions in the original code

- the initial aim is to get the code to work for default settings and then successively add different options. Since the code was discovered to be more complex than had initially been anticipated not all options may be incorporated, nevertheless, including them would be desirable.

(b) have a user guide for the program

- although the software will work without a guide, this requirement is desirable to use the software more effectively

(c) have a glossary for all of the parameters and variables used in the program

- although the program could be examined using the in-line comments in the code, a glossary would assist a programmer to be able to understand and modify the code more easily

3. Could:

(a) provide a GUI interface to plot the output results

- This is not required for the software to work effectively. The old program doesn't do this. Currently users of the old code simply import the output text files into Microsoft Excel in order to plot the results. However, this would be nice to have if possible.

4. Won't:

(a) include a web-based interface

- This might make the software even more available and accessible, however, it won't be possible to do this in this project. It is beyond the scope of this project. It is mentioned because it could possibly be considered as a future development

(b) be fully tested by several different users nor for a wide range of conditions

- that is, the number and type of tests will be limited due time constraints. The reason for this is due to difficulties in producing the software product. These are discussed later. They meant more time was spent on other aspects of the project.

3.3.2 Non-functional Requirements

The main non-functional requirements for this project are the following. Some functions which might otherwise be deemed non-functional in other systems, such as accessibility/availability, portability, being well-documented are considered functional in this project. Hence, it should be borne in mind that this requirements exist, but are considered functional here.

The system/product shall be:

1. Dependable/Reliable:
    - that is, the software should not generally not fail to produce output, i.e., should not fail to converge, more than the original code. Also the software note fail, but rather should alert the user if invalid input is provided.
2. Maintainable:
    - that is, the software should be easily maintainable by programmers, enabling them to quickly and easily o modify the code to correct mistakes or errors, or to add improvements to meet new requirements.
3. Usable:
    - that is, the software should be easy to learn to use by users and programmers. This means the interface should be intuitive and easy to use, with clear instructions and that the program documentation is helpful.
4. Testable:
    - that is, the system should be easy for a user to test the output results are correct. It should also be easy for a programmer to modify the code to carry out additional tests.

3.4 Use Cases

The use cases scenarios were determined from the current functionality of the old program. As already mentioned two types of actors are envisaged, users and programmers. To explain the use cases, a narrative description is given below on how each actor can use the system.

3.4.1 Use Cases Narrative Description

A user receives the software package along with the documentation. The user reads over the documentation. The user can use this to check the validity of the model. Then the user starts the software application. The user is presented with a GUI to set up the initial starting conditions of the experiment. The GUI shows the values of variables, some of which are set to default values and others of which are calculated. The user is able to change

some of the set variables. To effect the changes made by the user the user can instruct the software to recalculate all variables using the new input values. The user can accept the displayed variables as specifying the initial starting conditions. The user is presented with a GUI to set up and run a simulated experiment. The GUI shows the values of variables, some of which are set to default values and others of which are calculated. The user is able to change some of the set variables. The user is also able to choose from a range of options to alter the experimental conditions. The user can choose whether additional variables are required for output compared to the standard sets. The user chooses how the results should be output. The user can then instruct the program to run the simulated experiment.

A programmer can perform all of the steps that the user can perform. In addition, the programmer can also do the following steps. The programmer can read the source code for the software (which is commented) The programmer can use these to check the validity of the code. The programmer can also modify the code and run the modified code.

3.5 Analysis of Requirements

The main requirement for the system-to-be was that the software product, the computer program for the revised RBC model, gave results which match those of the old program. The correctness of software was checked at the software evaluation stage.

In order to ensure that the new program produced the same results as the old program it was necessary to determine the details of the revised RBC model and incorporate them into the program. This required, principally the deciphering of the old code using various resources. This is discussed further in Chapter 4.

As well as producing the new program an essential requirement was also to fully document the revised RBC model. This was done at the same time as creating the new program using the resources discussed in Chapter 4. The revised RBC model is documented in Chapter 5.

The new computer program for the RBC model was chosen to be implemented in the Java programming language. The reasons for this choice were the author's familiarity with this language, the presumed general familiarity of potential users of the program, and the fact that the language is platform-independent, is an object-oriented programming language, and because the language has a wide range of libraries and tools which could be used for the further development of the code.

In order that the code enable user input and also to be user-friendly it was decided to use a graphical user interface as part of the software. The graphical interface will use the Java swing libraries. To further enhance usability it was decided that the variables displayed in the GUI be given labels according to conventions in the research field. However, in the code itself they are given more descriptive labels to avoid any confusion as to their meaning.

The proposed interface is intended to be intuitive and user friendly. It should be noted that users of the software product, which are called "users" and "programmers" are assumed to be knowledgeable in the field of research of RBCs. An arbitrary non-specialist is not the intended target user of the software.

In order that the new code be maintainable the arrangement of variables and calculations in the code should be arranged in a logical manner. Furthermore, the sequence of steps in the code should be as logical and easy to follow as possible. To facilitate maintainability the code will be arranged into classes and objects that are logical.

On the topic of testability, standard Java programming practices will be followed to ensure testability. There will be separation of concerns, by the choices of appropriate classes and variable labels. The code will be well-commented so that the various parts of the code can be easy to follow. Moreover, the code will use of get and set methods for all variables so that they can be obtained and set safely for testing, etc.

The original old code and many details, including unpublished work were provided by the client. I am happy to provide the work reported in this project as a service to the client. Specifically, I do not claim any rights to copyright over the software product or documentation. I leave it to the client to decide on the copyright licensing.

The plan for implementing the functional requirements is given in System Design Stage described in Chapter 6.

While the requirements for this project can be specified and prioritised and later discussed how they will be implemented it is pertinent to consider whether there are any risks in this project. Upon careful consideration it is clear that there is a potential major risk in this project. Specifically, there is a risk that the details of the current old code cannot be deciphered. This would hinder documentation of the full model and creation of the new code. There is a risk that the deciphering may be difficult and more time-consuming than anticipated. This risk will affect the effectiveness of the whole project.

In order to determine the model and then document the model, the algorithms used in the code need to be determined. This was determined by a in stages using documentation and the old code. This is discussed in Chapter 4.

Chapter 4

Deciphering the Computer Program for the RBC Model

A major task in the project was to decipher the complete model as it now stands. To do this the following sources of information were used:

- the original RBC model of Lew and Bookchin (1986)
- the reticulocyte model of Lew et al. (1991)
- the unpublished draft manuscript by Weiss and Lew (2014) on a cellular homeostasis simulator
- the unpublished draft manuscript by Lew (2014a) on the calcium and magnesium extension
- the unpublished notes by Lew (2014c) on modelling cellular homeostasis
- a guide by Lew (2000) to using the BASIC program in practice
- the source code for the current BASIC computer program, called M1.BAS obtained from the client.

All of these sources of information were used together to determine the revised mathematical model. Specifically they were used to determine the equations in the model, the meanings of terms in those equations and the steps in the calculations. Once the details of the model are understood the software product, i.e., the new program, was created.

The difficulties a user may have in understanding and modifying the computer program for the RBC model have already been mentioned briefly. These are also pertinent to the difficulties for the analysis and deciphering of the program in this project. These and other difficulties with the program will now be discussed in more detail. Due to the difficulties mentioned below considerable time and effort in this project was expended deciphering the BASIC computer program for the RBC model.

4.1 Programming Language

The computer program for the RBC model is currently written in an old programming language. It was necessary for the author to learn this language, at least partially, to be able to understand the code.

4.2 Program Documentation

While there is some documentation explaining how to use the code and aspects of the model, such as Lew (2000), it is now out-of-date. It does not document the full model used by the code. This is because the code has undergone numerous modifications.

4.3 Comments in the Program

As has been mentioned earlier, the computer program is not particularly well-commented, particularly those parts relating to changes to the original model. This makes following the code difficult especially when calculations are carried out which do not seem to match the documentation.

4.4 Variable Labels in the Program

In this discussion the term variable is used to indicate a variable in a computer program. This can relate to a variable, parameter, or constant as used in the scientific sense.

The variables in the program are often not commented and, moreover, are given obscure labels such as A(1), A(2), A(6), B(5), B(7), B(8), I(25), I(52) etc. It was thus difficult to determine what the variables are in the code referred to in the model. Some variables are documented in the code or are given in a glossary to the program guide with more informative symbols enabling them to be inferred. However, many variables are not listed in that glossary. An example of uncommented, obscure labelling is the following at the beginning of the code:

```
'RS values of parameters and variables
370 A(1)=-10#:A(2)=6.450000000000004D-02:A(3)=.0258#:A(4)=7.2#
A(6)=0.01#:A(9)=34#
430 B(7)=1.5#:B(8)=37#:B(9)=4#:B(10)=2#
450 C(1)=10#:C(2)=140#:c(3)=95#:D(6)=0.001#:d(12)=0.0001#:E(8)=2d-10
490 M(1)=145#:M(2)=5#:M(3)=145#:M(5)=10#:M(7)=0#:M(9)=0#:M(10)=0#
i(77)=7.216#:i(78)=0.4#
```

Some of the variables like C(1), C(2), M(1), M(2) are defined in the aforementioned glossary, others are not. Some of the variables can be deduced from later information which is given

when values are printed out. Determining the meaning of the labels used in the code was not a trivial matter.

Another example of poorly commented labelling is the following:

```
'Default values if Mg screen is skipped
a23camk=10
a23mgmk=10
a23caik=10
a23mgik=10
mgt=2.5:q(7)=mgt
mgto=0.2
q0=0.05
atp=1.2
dpgp=15
V(2)=(1-A(11))+V(1)
gosub mgnr:'computes mgf in RS
pmg=0.01
```

Some of these are described later in the code:

```
input "MgoT? Default=0.2 mM. ENTER for default or change to:     ",mgto

. . . .

print "MgiT? (in mmole/loc); default=2.5.  This gives Mg2+ of about 0.3 "
print "mmole/lcw in oxygenated red cells with the default ATP+P and (2,3-DPG+P)
input "concentrations of 1.2 and 7.5 mmole/lc offered below.        ",mgt

. . . .

input "ATP+P? (in mmole/loc); ENTER for default=1.2:     ",atp

. . . .

input "2,3-DPG and other organic phosphates? (in mEq P/loc); ENTER for default=15:",dpgp
```

However, even with this information, the meaning of the variables is not immediately clear to a non-expert.

4.5 Structure of the Program

The structure of the code is not straightforward, with many jumps to different parts of the code with GOTO and GOSUB commands. This makes following the logical sequence difficult. This fragment of code provides an example of this difficulty.

```
2810 IF I(58)=5 THEN GOTO 5750

. . . .

2850 '.................Experimental changes begin here.....................
2870 CLS 2890 PRINT "1. TIME FACTORS.":PRINT
2910 IF (T(1)*60)=0 THEN GOTO 2930 ELSE GOTO 2990
2930 PRINT "Type a new value then 'Enter' (or 'Enter' alone for default values)."
2950 PRINT
2970 INPUT "LENGTH OF EXPERIMENT in minutes:      ",T(4):PRINT
2990 IF (T(1)*60)=0 THEN GOTO 3030 ELSE GOTO 3010

. . . .

5750 CLS 5770 IF R=2 THEN GOTO 16230
5790 IF T(4)=0 THEN 5810 ELSE 5850
5810 PRINT "YOU MUST ENTER THE LENGTH OF THE EXPERIMENT!"
5830 GOTO 2970
5850 PRINT "After"; T(4); "minutes will you wish to make further changes, Y/N";
5870 INPUT A$: PRINT:
5890 PRINT "Satisfied with entries (ENTER) or start again (1)?:  "
5910 INPUT "NB. Entering 1 will rerun the reference state!  ",I(49)
5930 IF I(49)=1 THEN GOTO 250
```

4.6 Modifications to the Program

As previously mentioned the code has been developed over time. Some previously used equations and assumptions were modified, and additional effects were also incorporated. Direct comparison with, say, the original model of Lew and Bookchin (1986) did not always help clarify the meaning of code. If after examining all of the documentation and the code the code remained unclear (electronic) communication was made with Prof. Lew to help clarify the modifications in the code.

```
810 'Protonized buffer concentration
pkhepes=7.83-0.014*b(8)
a(5)=10^(-pkhepes)
```

Here a(5) is the dissociation constant of the extra-cellular H-buffer, K_B, and rather than being given as a constant in the original model, is now expressed as a function of the temperature (in Celsius), $b(8)$.

4.7 Equations used in the Program

4.7.1 The Subject of the Equations

Although the RBC model of Lew and Bookchin (1986) outlines the steps in the computations in that model, in many cases the subject of the equations had to be changed to be able to use those equations as specified. Although not difficult, this time-consuming process had to be carried out by the author before some of the equations in some steps could be expressed explicitly. A full account of the algorithms used in the revised code, along with the explicit equations used is given later in this report.

4.7.2 Different Forms of the Equations

Different forms of the Equations used the Program

Furthermore, some of the equations in the code, although they have all of the variables expected, the equations appear to be different from those in the documentation. Upon further examination this was determined to be due to the ability to express the equations in different ways. For example, from Lew and Bookchin (1986) there is the coefficient

$$\frac{(C^c - C^m e^{-k})}{(1 - e^{-k})} \tag{4.1}$$

in the calculation of one of the types of fluxes but in the computer program the coefficient is written as

$$\frac{(C^m - C^c e^{k})}{(1 - e^{k})} \tag{4.2}$$

These are, in fact, entirely equivalent. However, the point here is that all the equations in the code had to be checked and compared with the equations in the papers. And any apparent differences checked and resolved.

4.7.3 Alternative Equations

In some cases, different equations were used to calculate some of the variable. Sometimes this was due to the fact that there was more than one equation already referred to in the model which could be used, in other cases it was because the new equations were added to the model. One example concerned the calculations of some of the fluxes. According to the paper, the fluxes were to be calculated from the equation for total fluxes, whereas in the code the fluxes were calculated from their own equations. This was not incorrect, in principle, but the equations used did differ.

4.8 Algorithms used in the Program, etc.,

The order of some steps in the procedures were found to be different from the documentation. This was determined not from the code but simple logic. For example, the initial

osmotic coefficient of haemoglobin, f^o_{Hb}, needs to be calculated before the quantity of the impermeant intracellular anion, Q_X, rather than the other way around as given in steps proposed in Lew and Bookchin (1986).

In addition, the sequence of some of the steps in the procedures are differed from that in the code. Furthermore, additional steps were included in some places in the code, apparently due to the inclusion of additional effects.

4.9 Magic Numbers in the Program

The code contained numerous examples of magic numbers, that is numbers used in the code which would have been better given with named variables are constants.

Some examples include the following:

```
V(6)=-(8.615600000000004D-02)*(273#+B(8))*LOG(V(7))
```

where the first term is due to the ideal gas constant divided by the Faraday constant.

```
mgb=mgb0+((atp/v(2))*mgf/(0.08+mgf))+((dpgp/v(2))*mgf/(3.6+mgf))
```

where 0.08, 3.6 are dissociation constants and should better be expressed using variables.

In addition some of the equations had explicit coefficients relating to the valences of the ions, such as +2, -2, +1, and -1. Many of the equations in the published papers do not include explicit mention of the valences of the ions in the equations - just the use of magic numbers. However, the draft by Weiss and Lew (2014) was helpful in clarifying the verbose forms of some equations.

4.10 Bugs in the Program

There appear to be some (presumably minor(?)) bugs in the old computer program.

For example, although the medium concentrations of Na^+ and anion, A^-, given by C^m_{Na} and C^m_A are set early in the code, they are (presumably inadvertently) adjusted later due to code that was used by Prof. Lew while experimenting about adding some effects to account for the adjustments of pH (Lew, private communication). Consultations with him revealed that such changes should not be made at that stage in the code. Although these adjustments only change the values slightly, they are, nonetheless, incorrect and confusing.

Another possible bug is the inclusion of the total quantity of Mg (which has bound and free components) along with the quantity of the free ions of Mg in the equation for the sum of quantities in the medium. This appears to be inconsistent with the closely related sum of concentrations in the medium where only the free ions of Mg are included.

4.11 Recalculations in the Program

The way the code is designed it seems to unnecessarily recalculate some variables and parameters. For example, some of the initial values for the total fluxes, such as Φ_{Na}, Φ_K, and Φ^{Co}, are set in the initial conditions, but the very same fluxes are later recalculated from other variables. These changes do not significantly change the values of the fluxes. Nevertheless, this led to some confusion as to what the code was doing. While the step with the calculations for such fluxes is appropriate after the initial conditions it is unnecessary for setting/calculating the initial conditions.

Chapter 5

The Revised Human Red-Blood Cell (RBC) Model

For simplicity the system is assumed to consist of two distinct regions, (the interior of) a single red-blood cell and the extra-cellular fluid in which it is suspended. These are referred to as the cell and the medium. The conditions, properties and substances contained within each of these regions can, in principle, vary. The two regions are separated by the cell membrane which facilitates the transportation of substances between the two regions.

5.1 The Equations used in the Model

The equations used in the Human Red-Blood Cell (RBC) Model are now considered in detail.

5.1.1 Equations from Conservation Laws

The law of conservation of mass leads to the conservation of water volume.

Conservation of Water Volume

Water is the solvent inside the cell and the medium. The passive diffusion of water is osmosis and it depends on the number of solute particles. The total osmotic pressure of the solution in each compartment is assumed to be the sum of the osmotic pressure contributions of all the component solutes in the solution. The osmotic pressure of the cell solution is given by

$$\Pi^c = \Sigma_k \Pi_k^c \tag{5.1}$$

where Π^c is the total osmotic pressure of the cell solution, Π_k^c is the osmotic pressure component of the solution for solute k in the cell water, and Σ_k indicates summation over all relevant solutes in the cell.

Similarly,

$$\Pi^m = \Sigma_j \Pi_j^m \tag{5.2}$$

where Π^m is the total osmotic pressure of the medium solution, Π_j^m is the osmotic pressure component of the solution for solute j in the medium water, and Σ_j indicates summation over all relevant solutes in the medium.

Conservation of water volume implies that $\Pi^c = \Pi^m$. This leads to

$$\Sigma_k \Pi_k^c = \Sigma_j \Pi_j^m \tag{5.3}$$

All relevant solutes for each region (i.e., cell or medium) need to be included in the above equation. In particular, solutes that are transportable between the regions (termed permeant) and those that are not transportable (impermeant) must be included. The permeant ionic solutes are: Na^+, K^+, Ca^{2+}, Mg^{2+}, H^+, and OH^- and A^-, where A^- stands for a permeant anion (such as chloride (Cl^-) or hydrogen carbonate (bicarbonate) (HCO_3^-)) and the other symbols represent the elements. Generally speaking, A^- can be assumed to be the same as Cl^-. The above permeant ionic solutes have concentrations both in the cell and in the medium.

Impermeant solutes for the cell include Hb ad X, where Hb stands for the hemoglobin molecule and X stands for an impermeant intracellular anion that is not protonizable (that is, unable to add a proton or H+ ion).

As for the medium, one impermeant solute is, B_T, the total (impermeant) extracellular H-buffer. This actually comprises two components a) a bound form, B_B and b) a free form, B_F^-. The free form is a monovalent anion. Other impermeant solutes include Y^-, an impermeant extracellular monovalent anion (such as the gluconate ion (C_6 H_{11} O_7^-)), Y^+, an impermeant extracellular monovalent cation (such as the glucamine ion(C_6 H_{15} N O_5^+)), Y^0, an impermeant extracellular electrically neutral molecule (such as sucrose (C_{12} H_{22} O_{11})).

Furthermore, magnesium and calcium can occur in bound form as well as the free-form of bivalent cations already mentioned. These forms can exist in both the cell and the medium. The total magnesium and calcium are indicated by Mg_T and Ca_T, while the bound forms are indicated by Mg_B and Ca_B. To avoid confusion the free-forms will sometimes be referred to as Mg_F and Ca_F instead of Mg^{2+} and Ca^{2+}.

Furthermore, Ca^{2+} ions within the cell can be studied by the introduction of a substance called benz2. The latter is a Ca^{2+} chelator (i.e., a substance that binds the Ca^{2+} ions). It has an effective electrical charge of -2. In this report benz is used to indicate benz2. Thus Q_{benz}^c stands for the quantity (amount) of benz2 in the cell.

Ethylene glycol tetraacetic acid (EGTA) and ethylene diamine tetraacetic acid (EDTA) are ligands, that is, molecules which bond to a central metal ion. They are useful as chelating agents for Ca^{2+} ions in the extra-cellular fluid. They each have an effective electrical charge of -1. Since only one of these will be used at any given time, in this report EDGTA is used to indicate either EGTA or EDTA.

Water molecules (H_2O) can dissociate into molecular hydroxide ions (OH^-) and hydrogen ions (H^+):

$$H_2O \leftrightarrow H^+ + OH^- \tag{5.4}$$

For aqueous media the equilibrium condition for water dissociation in the physical conditions applicable to cells in vivo is with $C_H^r C_{OH}^r = 10^{-14}$ M where C_H^r, C_{OH}^r are the concentrations of solutes H and OH in region r (in mole (M)). Neutrality occurs with $C_H^r = C_{OH}^r = 10^{-7}$ M $= 10^{-4}$ mM. These concentrations are much smaller than the concentrations of other relevant solutes, as well as the numerical computational errors of the other terms. Consequently, H^+ and OH^- may be ignored in the osmotic balance equation (also referred to as the isotonicity equation).

Reiterating, the relevant solutes for the cell, for the purposes of osmotic pressure balance, are (i.e., $k =$) Na^+, K^+, Ca^{2+}, Mg^{2+}, A^-, $benz^{2-}$, and Hb, and X, (the electrical charge of the last two can vary). The relevant solutes for the medium, for the purposes of osmotic pressure balance, are (i.e., $j =$) Na^+, K^+, Ca^{2+}, Mg^{2+}, A^-, B_T, Y^-, Y^+, Y^0, and $EDGTA^-$. Note $B_T = B_B + B_F^-$.

Henceforth, k and j will refer to the solutes but without explicit indication of the electrical charge, except for the Y solutes. Thus the relevant values for k are $k =$ Na, K, Ca_F, Mg_F, A, Hb, X, benz, and for j are $j =$ Na, K, Ca_F, Mg_F, A, B, Y^-, Y^+, Y^0, EDGTA.

The osmotic pressure of an ideal solution with low concentration can be derived from van't Hoff's equation (also known as the Morse equation):

$$\Pi_s^r = C_s^k RTi \tag{5.5}$$

where Π_s^r is the osmotic pressure of solute s in region r, C_s^r is the concentration of solute s in region r (in moles/L), R is the ideal gas constant (8.314 J K^{-1}), T is the absolute temperature (in Kelvin), and i is the dimensionless van't Hoff factor which is the number of ions or molecules into which the dissolved species dissociates. This equation is only valid for very dilute solutions. The equation can, however, be extended to non-ideal solutions and for more concentrated solutions using correction factors called osmotic coefficients. Thus the equation above becomes

$$\Pi_s^r = f_s C_s^r RTi \tag{5.6}$$

where f_s is the osmotic coefficient for solute s.

It should also be noted that the osmotic coefficient of hemoglobin unbound to protons is assumed the same as that for hemoglobin bound to protons.

So from equations (5.3) and (5.6)

$$\begin{aligned} f_{Na}Q_{Na}^c + f_K Q_K^c + f_{Mg_F}Q_{Mg_F}^c + f_{Ca_F}Q_{Ca_F}^c + f_A Q_A^c + f_{Hb}Q_{Hb}^c + f_X Q_X^c + f_{benz}Q_{benz}^c = \\ V_w^c(f_{Na}C_{Na}^m + f_K C_K^m + f_{Mg_F}C_{Mg_F}^m + f_{Ca_F}C_{Ca_F}^m + f_A C_A^m + f_B C_B^m + f_{Y^-}C_{Y^-}^m + f_{Y^+}C_{Y^+}^m + \\ f_{Y^0}C_{Y^0}^m + f_{EDGTA}C_{EDGTA}^m) \end{aligned} \tag{5.7}$$

where Q_k^c is the amount (i.e., quantity) of solute k in one litre of original packed cells, which is given by

$$Q_k^c = V_w^c \times C_k^c \tag{5.8}$$

where C_k^c is the concentration of solute k in the cell and V_w^c is the volume of cell water.

Usually, the osmotic coefficient of all solutes bar hemoglobin are assumed to be unity. Thus the equation for osmotic pressure is thus given by:

$$\begin{aligned} &Q^c_{Na} + Q^c_K + Q^c_{Mg_F} + Q^c_{Ca_F} + Q^c_A + f_{Hb}Q^c_{Hb} + Q^c_X + Q^c_{benz} \\ &= V^c_w(C^m_{Na} + C^m_K + C^m_{Mg_F} + C^m_{Ca_F} + C^m_A + C^m_B + C^m_{Y^-} + C^m_{Y^+} + C^m_{Y^0} + C^m_{EDGTA}) \end{aligned} \quad (5.9)$$

Following McConaghy and Maizels (1961) and Freedman and Hoffman (1979) the osmotic coefficient of hemoglobin is given by the empiral equation

$$f_{Hb} = 1 + b\frac{Q^c_{Hb}}{V^c_w} + c\left(\frac{Q^c_{Hb}}{V^c_w}\right)^2 \quad (5.10)$$

where b and c are virial coefficients.

Equation (2.9) can also be re-written as

$$f_{Hb} = 1 + bC^c_{Hb} + c\left(C^c_{Hb}\right)^2 \quad (5.11)$$

where $C^c_{Hb} = Q^c_{Hb}/V^c_w$.

Conservation of Electroneutrality

The electrical charge of each region (cell or medium) must remain neutral. That is,

$$\Sigma_k q^c_k = 0 \quad (5.12)$$

and

$$\Sigma_j q^m_j = 0 \quad (5.13)$$

where q^c_k is the total electrical charge of solute k within the cell and q^m_j is the total electrical charge of solute j in the medium.

The total electrical charge of solute i in the cell is given by

$$q^c_k = z_k Q^c_k \quad (5.14)$$

where z_k is the valence (i.e., mean net electrical charge number) of solute k and Q^c_k is the amount (i.e., quantity) of solute k in one litre of original packed cells.

As before we will ignore the contributions from solutes H^+ and OH^-. Thus charge neutrality within the cell is given by

$$z_{Na}Q^c_{Na} + z_K Q^c_K + z_{Mg}Q^c_{Mg} + z_{Ca}Q^c_{Ca} + z_A Q^c_A + z_{Hb}Q^c_{Hb} + z_X Q^c_X + z_{benz}Q^c_{benz} = 0 \quad (5.15)$$

From Lew and Bookchin (1986) the valence (or mean net electrical charge) of hemoglobin is given by

$$z_{Hb} = a(pH^c - pI^c) \quad (5.16)$$

where a is the coefficient representing the slope of the titration curve (in mEq/mmol), pH^c is the cell pH, and pI^c is the cell pH when $z_{Hb} = 0$.

Electroneutrality in the cell means the valence of the impermeant intracellular anion, X, can be determined from:

$$z_X = -\frac{(z_A Q_A^c + z_{benz} Q_{benz} + z_{Na} Q_{Na}^c + z_K Q_K^c + z_{Mg_F} Q_{Mg}^c + z_{Ca_F} Q_{Ca}^c + z_{Hb} Q_{Hb}^c)}{Q_X^c} \tag{5.17}$$

5.1.2 Equations that Define Specific Membrane Transport Mechanisms

Membrane transport processes may be divided into two broad categories, namely, active transport and passive transport. Transport processes across the cell membrane are selective and only transport a specific subset of solutes.

Passive transport involves three types of diffusion, namely, simple diffusion, channel-mediated diffusion, and carrier-mediated diffusion. The latter two are both also considered to be facilitated diffusion. In each of the different types of passive transport the solute is transported from the region of high concentration to the region of low concentration.

For active transport the processes are pump-mediated. The pump on the cell membrane transports one or more specific solutes from the region of low concentration to the region of high concentration for that solute, i.e., *against* the solute concentration gradient. The pump uses the energy released by the breakdown of adenosine triphosphate (ATP) molecules within the cell to effect the transportation.

In the case of simple diffusion small uncharged polar molecules like water (H_2O) as well as lipid-soluble (hydrophobic) molecules (such as O_2, CO_2) can diffuse across the cell membrane. However, large uncharged polar molecules, such as sucrose, as well as (electrically charged) ions such as Na^+, K^+, Ca^{2+}, Mg^{2+}, and Cl^- cannot transport by simple diffusion.

Facilitated diffusion is where diffusion is helped by transport proteins in the cell membrane. There are two types, carriers and channels. These can be used to transport solutes in the direction of the solute concentration gradient.

The total flux of the solute s through the cell membrane is given by

$$\Phi_s = \Sigma_M \Phi_s^M \tag{5.18}$$

where Φ_s is the total flux of the solute s through the cell membrane and Φ_s^M is the flux of the solute s through the cell membrane for cell membrane transport mechanism M. The summation is over all transport mechanisms. The relevant cell membrane transport mechanisms, M, differ for each solute s, as described below.

The original model of Lew and Bookchin (1986) modelled "passive" transport by leaks, or low-saturation facilitated diffusion, however the code was later upgraded to model this instead as due to co-transport with Cl^-, referred to here as NaA cotransport and KA cotransport, where, as before, A is taken to be an anion (typically Cl^-).

The different transport processes across the cell membrane will now be discussed briefly.

The sodium-potassium pump

The flux of sodium through the sodium-potassium pump is given by

$$\Phi_{Na}^{P} = -\Phi_{Na}^{\max}\left[\frac{C_{Na}^{c}}{\left(C_{Na}^{c} + K_{m_{Na}}\left[1 + \frac{C_{K}^{c}}{K_{I_K}}\right]\right)}\right]^{3}\left[\frac{C_{K}^{m}}{\left(C_{K}^{m} + K_{m_K}\left[1 + \frac{C_{Na}^{m}}{K_{I_{Na}}}\right]\right)}\right]^{2} \tag{5.19}$$

where $\Phi_{Na}^{\max}$ is the sodium efflux through the sodium-potassium pump saturated with internal sodium and external potassium at the ATP levels of normal RBCs, and where coefficients $K_{m_{Na}}$, K_{I_K}, K_{m_K}, $K_{I_{Na}}$, represent the dissociation constant for Na activation of the sodium-potassium pump in the cell, the dissociation constant for K inhibition of the sodium-potassium pump in the cell, the for dissociation constant for K activation of the sodium-potassium pump in the medium, and the for dissociation constant for Na inhibition of the sodium-potassium pump in the medium.

The sodium-potassium pump pumps 3 Na^+ ions out of the the cell while pumping 2 K^+ ions into the the cell. Thus the flux of potassium through the sodium-potassium pump as

$$\Phi_{K}^{P} = -\frac{2}{3}\Phi_{Na}^{P} \tag{5.20}$$

The Calcium Pump

Usually there is a large transmembrane electrochemical gradient of Ca^{2+} which drives Ca^{2+} into the cell. However, the Calcium (Ca^{2+}) pump is a transport protein in the cell membrane which counters this effect by actively transporting Ca^{2+} out of the cell.

Goldman-type electrodiffusion

Goldman-type electrodiffusion refers to facilitated diffusion via electrodiffusional channels. That is the ions diffuse across the cell membrane due the presence a voltage (or an electrical potential difference) across the cell membrane. The diffusion depends on the electrical potential across the membrane as well as the concentrations of ions inside and outside of the cell.

The flux equation for Goldman-type electrodiffusion is given by

$$\Phi_{i}^{G} = P_{i}^{G} z_i \frac{EF^2}{RT}\left(\frac{C_{i}^{c} - C_{i}^{m}\exp\left(-\frac{z_i EF}{RT}\right)}{1 - \exp\left(-\frac{z_i EF}{RT}\right)}\right) \tag{5.21}$$

where Φ_{i}^{G} is the electrodiffusional flux for solute i, P_{i}^{G} is the membrane permeability constant for solute i through electrodiffusional pathways (in m s^{-1}). E is the transmembrane electrical potential (in V), and F is the Faraday constant (9.64853×10^4 C mol^{-1}). Here i = Na, K, A, H.

Na:K:2A cotransport

The sodium-potassium-chloride (Na^+-K^+-$2Cl^-$) cotransporter is a membrane protein that transports two Cl^- ions along with one Na^+ and one K^+ ions into the cell.

The flux of solute Na^+ or K^+ via the sodium-potassium-chloride cotransporter is given by

$$\Phi^{Co} = -k^{Co}\left[(C_A^c)^2\, C_{Na}^c C_K^c - d\,(C_A^m)^2\, C_{Na}^m C_K^m\right] \tag{5.22}$$

where k^{Co} is the turnover rate constant for the Na-K-2A cotransport. We have $\Phi_A^{Co} = 2\Phi_{Na}^{Co} = 2\Phi_K^{Co}$.

Na:A cotransport

The sodium-chloride Na^+-Cl^- cotransporter is a membrane transport protein that uses the strong Na^+ electrochemical gradient into the cell to transport a Cl^- ion into the cell along with each Na^+ ion.

The Na flux for this transporter is given by:

$$\Phi_{Na}^{NaA} = -\frac{k^{NaA}}{f_T^{pass}}\,(C_{Na}^c C_A^c - C_{Na}^c C_A^c) \tag{5.23}$$

where k^{NaA} is the turnover rate constant for the NaA cotransporter and f_T^{pass} is the temperature dependent factor for passive transport. The latter term is given by $f_T^{pass} = \exp((T_C^o - T_C)/10)\log(Q_{10}^{pass})$ where T_C is the temperature in Celsius, T_C is the initial temperature in Celsius, and Q_{10}^{pass} is the $Q_{10}$of passive permeability pathways.

K:A cotransport

The potassium-chloride (K^+-Cl^-) cotransporter is a membrane transport protein that uses the large K^+ electrochemical gradient out of the cell to transport a Cl^- ion out of the cell along with each K^+ ion.

The K flux for this transporter is given by:

$$\Phi_K^{KA} = -\frac{k^{KA}}{f_T^{pass}}\,(C_K^c C_A^c - C_K^c C_A^c) \tag{5.24}$$

where k^{KA} is the turnover rate constant for the KA cotransporter.

The flux of anions, A, for the NaA and KA transporters is given by:

$$\Phi_A^{NaAKA} = \Phi_{Na}^{NaA} + \Phi_K^{KA} \tag{5.25}$$

H:A cotransport due to the Jacobs-Stewart cycle

This is facilitated diffusion via the Jacobs-Stewart cotransporter.

The flux of solute i via the Jacobs-Stewart cotransporter is given by

$$\Phi_i^{HA} = -k_{HA}\left(C_A^c C_H^c - C_A^m C_H^m\right) \tag{5.26}$$

where i = H, A, and k_{HA} is the turnover rate constant for HA cotransport. We have $\Phi_H^{HA} = \Phi_A^{HA}$.

Total Fluxes

The equations for the total fluxes of Na^+, K^+, Ca^{2+}, Mg^{2+}, A^-, and H^+ ions across the cell membrane are given below. In these equations the subscripts indicate the ion species transported and the superscripts indicate the transport process. (An ionophore is a molecule which can traverse and transport an ion across the cell membrane). In the equations given below the superscripts indicate the following:

NaKPump	the sodium-potassium pump
CaPump	the calcium pump
CaPassive	passive calcium transport
A23187Mg2H	transport of Mg and H due to the ionophore A23187
A23187Ca2H	transport of Ca and H due to the ionophore A23187
G	Goldman-type electrodiffusion
NaK2A	NaK2A cotransport
NaA	Na:A cotransport
KA	K:A cotransport
HA	H:A cotransport due to the Jacobs-Stewart cycle

The total fluxes in the system are given by

$$\Phi_{Na} = \Phi_{Na}^{NaKPump} + \Phi_{Na}^{G} + \Phi_{Na}^{NaK2A} + \Phi_{Na}^{NaA} \tag{5.27}$$

$$\Phi_{K} = \Phi_{K}^{NaKPump} + \Phi_{K}^{G} + \Phi_{K}^{NaK2A} + \Phi_{K}^{KA} \tag{5.28}$$

$$\Phi_{A} = \Phi_{A}^{HA} + \Phi_{A}^{G} + \Phi_{A}^{NaK2A} + \Phi_{A}^{NaA} + \Phi_{A}^{KA} \tag{5.29}$$

$$\Phi_{H} = \Phi_{H}^{HA} + \Phi_{H}^{G} + \Phi_{H}^{CaPump} + \Phi_{H}^{A23187Mg2H} + \Phi_{H}^{A23187Ca2H} \tag{5.30}$$

$$\Phi_{Ca} = \Phi_{Ca}^{CaPump} + \Phi_{Ca}^{CaPassive} + \Phi_{Ca}^{A23187Ca2H} \tag{5.31}$$

$$\Phi_{Mg} = \Phi_{Mg}^{A23187Mg2H} \tag{5.32}$$

5.2 The Procedure for using the equations in the RBC Model

The code consists of these parts:

- Setting up the initial starting state, called the reference state
- The dynamic state involves
  - calculating the integration interval (which is based on the current values of the fluxes)
  - using that to calculate the small changes in the fluxes and concentrations for the next step
  - calculating the new values of the fluxes and concentrations for the next step
  - repeating the process until the experiment ends
  - printing the results to output files

The dynamic state is the main loop of the code. It is where perturbations occur and changes to the fluxes and concentration are calculated.

In order to develop an understanding of the variables and how they are calculated a series of algorithms was developed.

The first algorithm was used to calculate parameters and variable for the reference state as described in the paper. The procedure required that some equations had to change the subject of the equations. So, explicit mention is given of every equation with the correct subject.

The second algorithm was created based on the procedure to calculate the reference state in the old code. This algorithm was broken up into a set of algorithms, but the steps were essentially the same as in the old code.

Finally, a third algorithm was created based on the software design chosen for the new computer program. From that algorithm the new computer program can be created.

5.3 The Algorithm used to calculate the Reference State of the RBC Model in Lew and Bookchin (1986)

Here I follow the steps in the computation of the reference state given in the article Lew and Bookchin (1986).

To aid my understanding and to emphasize the logical structure, I only mention variables prior to when they are required rather than define all of them at the beginning. When the preliminary new code is written these initially set variables are indeed all set at the beginning. The purpose of setting out this algorithm is as the first step in defining the algorithm as defined in the old program.

The steps in the algorithm consist of set commands where variables are set to values and calculate commands where the variables are calculated from the equations in the model. The equations are given explicitly in the algorithm.

I made some small changes. I modified the equations given in Lew and Bookchin (1986) to explicitly mention the valences of the ions. I also used the symbol z to indicate valences, rather than n as in the paper. Moreover, in many cases I also use an explicit superscript c to indicate that the variable is for the cell, even when this is obvious, such as for the various quantities, Q. This is done merely for emphasis. I also use uppercase phi Φ rather than lowercase phi ϕ to represent the fluxes.

This is one algorithm but it is split into two for readability, otherwise it will would go beyond the length of the page.

The algorithm to determine the reference state according to the paper of Lew and Bookchin (1986) is shown below. The first part is here:

The second part to the algorithm to determine the reference state according to the paper of Lew and Bookchin (1986) is shown below:

Algorithm 1b Compute the initial reference state from paper only (part 2 of 2)

1: **procedure** REFERENCESTATE(B)
2: Set initial cell value for $\Phi^P_{Na} = 2.6$
3: Calculate initial value for $\Phi^P_K = \Phi^P_{Na}/1.5$
4: Set initial values for constants $k_1 = 0.2$, $k_2 = 8.3$, $k_3 = 0.1$, $k_4 = 18$
5: Calculate initial value for $\Phi^{max}_{Na} = -\dfrac{\Phi^P_{Na}}{\left[\dfrac{C^c_{Na}}{\left(C^c_{Na}+k_1\left[1+\frac{C^c_K}{k_2}\right]\right)}\right]^3\left[\dfrac{C^m_K}{\left(C^m_K+k_3\left[1+\frac{C^m_{Na}}{k_4}\right]\right)}\right]^2}$
6: Set initial $f_G = 0.1$ or 0.9
7: Set initial values for $\Phi^{Co} = \Phi^{Co}_{Na} = \Phi^{Co}_K = \Phi^{Co}_A = 0$
8: **for** each ion i = Na, K **do**
9: Set initial $\Phi_i = 0$
10: Calculate initial value for $\Phi^G_i = \dfrac{\left(\Phi_i - \Phi^P_i - \Phi^{Co}_i\right)}{\left(1+\frac{(1-f_G)}{f_G}\right)}$
11: Calculate initial value for $P^G_i = -\left(\dfrac{\Phi^G_i}{z_i\frac{EF}{RT}}\right)\dfrac{\left(1-\exp\left[-\frac{z_iEF}{RT}\right]\right)}{\left(C^c_i - C^m_i\exp\left[-\frac{z_iEF}{RT}\right]\right)}$
12: Calculate initial value for $\Phi^L_i = \Phi_i - \Phi^P_i - \Phi^{Co}_i - \Phi^G_i$
13: Calculate initial value for $P^L_i = -\dfrac{\Phi^L_i}{(C^c_i - C^m_i)}$
14: **end for**
15: Return from loop with Φ_{Na}, Φ^G_{Na}, P^G_{Na}, Φ^L_{Na}, P^L_{Na}, Φ_K, Φ^G_K, P^G_K, Φ^L_K, P^L_K
16: Set initial $k_{Co} = 10^{-9}$ or $= 10^{-6}$
17: Calculate initial value for $d = \dfrac{\left(\frac{\Phi^{Co}}{k_{Co}} + (C^c_A)^2 C^c_{Na} C^c_K\right)}{\left(C^m_A\right)^2 C^m_{Na} C^m_K}$
18: Set initial $k_{HA} = 1$ or $= 10^9$
19: Set initial P^G_A to be a value in the range 0.2–200.
20: **end procedure**

Algorithm 1a Compute the initial reference state from paper only (part 1 of 2)

procedure ReferenceState(A)
  Set initial experimental condition for $pH^m = 7.40$
  Calculate initial value for $C_H^m = 10^{-pH^m}$
  Set initial experimental conditions for $C_{BTotal}^m = 10, K_B = 10^{-7.55}$
  Calculate initial value for $C_{BBound}^m = C_{BTotal}^m \left(\frac{C_H^m}{K_B + C_H^m}\right)$
  Set final values for $z_{Na} = z_K = +1$; $z_A = z_B = z_{Y-} = -1$
  Set initial experimental conditions for $C_{Na}^m = 140, C_K^m = 5, C_{Y-}^m = 0$
  Calculate initial value for $C_A^m = z_{Na}C_{Na}^m + z_K C_K^m + z_B(C_B^m - C_{HB}^m) + z_{Y-}C_{Y-}^m$
  Set initial cell value for $C_A^c = 95$
  Calculate initial value for $r_A = \frac{C_A^m}{C_A^c}$
  Set initial value for $\Phi^{HA} = \Phi_A^{HA} = \Phi_H^{HA} = 0$
  Calculate initial value for r_H. From above $\Phi^{HA} = 0$, this implies that initial $r_H = r_A$
  Calculate initial value for $C_H^c = r_H C_H^m$
  Calculate initial value for $pH^c = -\log_{10}(C_H^c)$
  Define universal physical constants $R = 8.314462$, $F = 9.64853399 \times 10^4$.
  Set initial value for temperature T
  Calculate initial value for $E = -\left(\frac{RT}{F}\right) log\,(r_A)$
  Set initial cell values for $Q_{Hb}^c = 5$, $V_w^c = 0.7$
  Calculate initial value for $C_{Hb}^c = \frac{Q_{Hb}^c}{V_w^c}$
  Set initial cell value for $b = 0.0645, c = 0.0258$
  Calculate initial value for $f_{Hb} = 1 + bC_{Hb}^c + c\,(C_{Hb}^c)^2$
  Set initial cell values for $C_{Na}^c = 10, C_K^c = 140$
  for each ion i = Na, K, A **do**
    Calculate initial value for $Q_i^c = C_i^c V_w^c$
  end for
  Return from loop with Q_{Na}^c, Q_K^c, and Q_A^c
  Set initial cell value for $Q_{Mg}^c = 2.5$
  Calculate initial value for $Q_X^c = V_w^c(C_{Na}^m + C_K^m + C_A^m + C_B^m + C_{Y-}^m) - Q_{Na}^c - Q_K^c - Q_A^c - Q_{Mg}^c - f_{Hb}Q_{Hb}^c$
  Set initial cell value for $a = -10, pI^c = 6.85$
  Calculate initial value for $z_{Hb} = a(pH^c - pI^c)$
  Set final values for $z_{Mg} = +2$
  Calculate initial value for $z_X = -\frac{(z_A Q_A^c + z_{Na}Q_{Na}^c + z_K Q_K^c + z_{Mg}Q_{Mg}^c + z_{Hb}Q_{Hb}^c)}{Q_X^c}$
  Calculate initial value for $Q_{(-)}^c = z_{Hb}Q_{Hb}^c + z_X Q_X^c + z_{Mg}Q_{Mg}^c$
end procedure

5.4 The Algorithm used to calculate the Reference State of the RBC Model in the old program

The algorithm in the old program is more complicated than that above. Consequently, for readability and simplicity we split the algorithm up into many parts defined by different sections of the old program. However, it should be noted that together these all are actually one algorithm. The letters RS refer to the reference state. The algorithms range range from algorithm 2a – algorithm 2n.

Algorithm 2a Compute RS from code & paper: (part 01 of 14)

1: **procedure** RS(A)
2: Set final values for unchangeable constants: valences $z_{Mgfree} = z_{Cafree} = +2$; $z_{Na} = z_K = z_{Yglucamine} = +1$; $z_{HB} = z_{Ysucrose} = 0$; $z_A = z_{Bfree} = z_{Ygluconate} = -1$; $z_{benz2} = -2$; ideal gas constant $R = 8.314462$; Faraday constant $F = 9.64853399 \times 10^4$; fraction of Na-K transfer in Na-K pump, $sNaK = 3/2$.
3: **end procedure**

Algorithm 2b Compute RS from code & paper: (part 02 of 14)

1: **procedure** RS(B)
2: Set initial experimental condition for $pH^m = 7.40$
3: Calculate initial value for $C_H^m = 10^{-pH^m}$
4: Set initial experimental condition for temperature in Celsius $T_C = 37$
5: Calculate initial value for temperature in Kelvin $T_K = T_C + 273.15$
6: Calculate initial value for $pK_a^{HEPES} = 7.83 - 0.014T_C$
7: Calculate initial value for $K_B = 10^{-pK_a^{HEPES}}$
8: Set initial experimental condition for $C_B^m = 10$
9: Calculate initial value for $C_{HB}^m = C_B^m \left(\frac{C_H^m}{K_B + C_H^m}\right)$
10: Calculate initial value for $C_{Bfree}^m = C_B^m - C_{HB}^m$
11: **Set initial experimental condition for** $C_{Na}^m = 145$
12: Set initial experimental condition for $C_K^m = 5$
13: Set initial experimental condition for $C_{Ygluconate}^m = 0$
14: Set initial experimental condition for $C_{Yglucamine}^m = 0$
15: Set initial experimental condition for $C_{Ysucrose}^m = 0$
16: **end procedure**

5.5 The Algorithms used to calculate the RS for the new code

Below I present the algorithms used to calculate the RS for the new computer program. These are divided according the classes used in the software design discussed in Chapter 6. I had hoped to detail the equations explicitly but haven't managed to do that due to time constraints. For those reasons the algorithms for the Membrane for the new code are not given here although they were implemented in the new program. The new code contains the correct equations, but these have not been written down as algorithms. The algorithms range from algorithm 3 – algorithm 5.

5.6 The Algorithms used to calculate the Dynamic State for the new code

The Dynamic State was implemented in the new program. The algorithms for the Dynamic State are not listed here.

Algorithm 2c Compute RS from code & paper: (part 03 of 14)

procedure RS(C)
    Set initial value for MCHC = 34.
    Calculate initial value for $V_w^c = 1 - MCHC/136$
    Calculate initial value for $Q_{Hb}^c = MCHC \times 10/64.5$
    Calculate initial value for $C_{Hb}^c = \frac{Q_{Hb}^c}{V_w^c}$
    Set initial cell value for $C_{Na}^c = 10$
    Calculate initial value for $Q_{Na}^c = C_{Na}^c V_w^c$
    Set initial cell value for $C_K^c = 140$
    Calculate initial value for $Q_K^c = C_K^c V_w^c$
    Set initial value for Q10 of pumps, $Q10^P = 4$
    Set initial value for Q10 of passive permeability pathways, $Q10^L = 2$
    Set initial $f_G = 0.1$ or 0.9
    Set initial value for $vlysis = 1.45$
    Set initial cell value for $b = 6.45 \times 10^{-2}$
    Set initial cell value for $c = 2.58 \times 10^{-2}$
    Calculate initial value for $f_{Hb} = 1 + bC_{Hb}^c + c\left(C_{Hb}^c\right)^2$
end procedure

Algorithm 2d Compute RS from code & paper: (part 04 of 14) : Mg

procedure RS(D)
  Set initial experimental value for $C^m_{Mg} = 0.2$
  Set initial cell value for $Q^c_{Mg} = 2.5$
  Set initial experimental value for $C^m_{Mgfree} = C^m_{Mg} = 0.2$
  Set initial value for high-affinity buffer for Mg, $MgB_0 = 0.05$
  Set initial value for ATP+P for Mg, $MgB_1(= atp) = 1.2$
  Set initial value for 2,3-DPG and other organic phosphates for Mg, $MgB_2(= dpgp) = 15$
  Set initial value for $Kd^1_{Mg} = 0.08$
  Set initial value for $Kd^2_{Mg} = 3.6$
  Set initial value for $dMg = 1 \times 10^{-4}$
  Set initial value for $yMg = 1$
  Set initial value for $C^c_{Mgfree} = 2 \times 10^{-2}$
  Calculate initial value for C^c_{Mgfree} using this routine
  while $yMg > 1 \times 10^{-5}$ **do**
    Set $C^c_{Mgf1} = C^c_{Mgfree} - dMg$
    Calculate $b_1 = MgB_0 + (\frac{MgB_1}{vv})\frac{C^c_{Mgf1}}{(Kd^1_{Mg}+C^c_{Mgf1})} + (\frac{MgB_2}{vv})\frac{C^c_{Mgf1}}{(Kd^2_{Mg}+C^c_{Mgf1})}$
    Calculate $y_1 = Q^c_{Mg} - C^c_{Mgf1}\frac{Vw}{vv} - b_1$
    Set $C^c_{Mgf2} = C^c_{Mgfree} + dMg$
    Calculate $b_2 = MgB_0 + (\frac{MgB_1}{vv})\frac{C^c_{Mgf2}}{(Kd^1_{Mg}+C^c_{Mgf2})} + (\frac{MgB_2}{vv})\frac{C^c_{Mgf2}}{(Kd^2_{Mg}+C^c_{Mgf2})}$
    Calculate $y_2 = Q^c_{Mg} - C^c_{Mgf2}\frac{Vw}{vv} - b_2$
    Calculate $ss = (y_2 - y_1)/(C^c_{Mgf1} - C^c_{Mgf2})$
    Calculate $C^c_{Mgfree} = C^c_{Mgf1} - y_1/ss$
    Calculate $b_3 = MgB_0 + (\frac{MgB_1}{vv})\frac{C^c_{Mgfree}}{(Kd^1_{Mg}+C^c_{Mgfree})} + (\frac{MgB_2}{vv})\frac{C^c_{Mgfree}}{(Kd^2_{Mg}+C^c_{Mgfree})}$
    Calculate $yMg = Q^c_{Mg} - C^c_{Mgfree}\frac{Vw}{vv} - b_3$
  end while
  Return with recomputed value for C^c_{Mgfree}
  Set initial value for $PMg = 1 \times 10^{-2}$
  Set initial value for $A23Camk = 10$
  Set initial value for $A23Mgmk = 10$
  Set initial value for $A23Caik = 10$
  Set initial value for $A23Mgik = 10$
  Set initial value for $KD^a_{Mg} = 0.05$
  Set initial value for $KD^i_{Mg} = 4.$
  Calculate initial value for factor $Mgfactor = \left(\frac{C^c_{Mgfree}}{KD^a_{Mg}+C^c_{Mgfree}}\right)\left(\frac{KD^i_{Mg}}{KD^i_{Mg}+C^c_{Mgfree}}\right)$
end procedure

Algorithm 2e Compute RS from code & paper: (part 05 of 14) : Ca

procedure RS(E)
  Set initial experimental value for $C^m_{Ca} = 1$
  Set initial cell value for $Q^c_{Ca} = 5.8 \times 10^{-4}$
  Set initial experimental value for $C^m_{Cafree} = C^m_{Ca} = 1$
  Calculate initial value for $C^c_{Ca} = Q^c_{Ca}/V^c_w$
  Set initial value for α (= sort of inverse part of CaB_0) $= 0.30$
  Set initial value for $CaB_1 (= B1Ca) = 0.026$
  Set initial value for $CaB_2 (= benz2) = 0$
  Set initial value for $Kd^1_{Ca} (= Kd^{B1}_{Ca}) = 0.014$
  Set initial value for $Kd^2_{Ca} (= Kd^{benz2}_{Ca}) = 5 \times 10^{-5}$
  Set initial value for $dCa = 1 \times 10^{-6}$
  Set initial value for $yCa = 1$
  Set initial value for $C^c_{Cafree} = 1.12 \times 10^{-4}$
  Calculate initial value for C^c_{Cafree} using this routine
  Calculate $C^c_{Cafree} = 1000 \times C^c_{Cafree}$
  Calculate $Q^c_{Ca} = 1000 \times Q^c_{Ca}$
  Calculate $CaB_1 = 1000 \times CaB_1$
  Calculate $CaB_2 = 1000 \times CaB_2$
  Calculate $Kd^1_{Ca} = 1000 \times Kd^1_{Ca}$
  Calculate $Kd^2_{Ca} = 1000 \times Kd^2_{Ca}$
  while $yCa > 1 \times 10^{-6}$ **do**
    Set $C^c_{Caf1} = C^c_{Cafree} - dCa$
    Calculate $b_1 = \frac{C^c_{Caf1}}{\alpha} + (\frac{CaB_1}{vv})\frac{C^c_{Caf1}}{(Kd^1_{Ca}+C^c_{Caf1})} + (\frac{CaB_2}{vv})\frac{C^c_{Caf1}}{(Kd^2_{Ca}+C^c_{Caf1})}$
    Calculate $y_1 = Q^c_{Ca} - b_1$
    Set $C^c_{Caf2} = C^c_{Cafree} + dCa$
    Calculate $b_2 = \frac{C^c_{Caf2}}{\alpha} + (\frac{CaB_1}{vv})\frac{C^c_{Caf2}}{(Kd^1_{Ca}+C^c_{Caf2})} + (\frac{CaB_2}{vv})\frac{C^c_{Caf2}}{(Kd^2_{Ca}+C^c_{Caf2})}$
    Calculate $y_2 = Q^c_{Ca} - b_2$
    Calculate $ss = (y_2 - y_1)/(C^c_{Caf1} - C^c_{Caf2})$
    Calculate $C^c_{Cafree} = C^c_{Caf1} - y_1/ss$
    Calculate $b_3 = \frac{C^c_{Cafree1}}{\alpha} + (\frac{CaB_1}{vv})\frac{C^c_{Cafree}}{(Kd^1_{Ca}+C^c_{Cafree})} + (\frac{CaB_2}{vv})\frac{C^c_{Cafree}}{(Kd^2_{Ca}+C^c_{Cafree})}$
    Calculate $yCa = Q^c_{Ca} - b_3$
  end while
  Calculate $C^c_{Cafree} = C^c_{Cafree}/1000$
  Calculate $Q^c_{Ca} = Q^c_{Ca}/1000$
  Calculate $CaB_1 = CaB_1/1000$
  Calculate $CaB_2 = CaB_2/1000$
  Calculate $Kd^1_{Ca} = Kd^1_{Ca}/1000$
  Calculate $Kd^2_{Ca} = Kd^2_{Ca}/1000$
  Return with recomputed value for C^c_{Cafree}
    where $vv = V^c_w + \frac{MCHC}{136}$
  Set initial value for $fcapm = 12$
  Set initial value for $capk = 2 \times 10^{-4}$
  Set initial value for $h1 = 4$
  Set initial value for $cah = 0 or 1 or 2$
  Set initial value for $hik = 4 \times 10^{-7}$
  Set initial value for $capmgk = 0.1$
  Set initial value for $fcalm = 0.05$
  Set initial value for $pkcak = 1 \times 10^{-2}$
end procedure

Algorithm 2f Compute RS from code & paper: (part 06 of 14) : Anion

procedure RS(F)
  Calculate initial value for $C_A^m = (z_{Na}C_{Na}^m + z_K C_K^m + z_{Mgfree}C_{Mgfree}^m + z_{Cafree}C_{Cafree}^m + +z_{Yglucamine}C_{Yglucamine}^m - z_{Bfree}C_{Bfree}^m - z_{Ygluconate}C_{Ygluconate}^m)/(-z_a)$
  Set initial cell value for $C_A^c = 95$
  Calculate initial value for $Q_A^c = C_A^c V_w^c$
  Calculate initial value for $r_A = \frac{C_A^m}{C_A^c}$
end procedure

Algorithm 2g Compute RS from code & paper: (part 07 of 14) : Hydrogen

procedure RS(G)
  Calculate initial value for r_H. From above $\Phi^{HA} = 0$, this implies that initial $r_H = r_A$
  Calculate initial value for $C_H^c = r_H C_H^m$
  Calculate initial value for $pH^c = -\log_{10}(C_H^c)$
  Set initial cell value for pI^c at temperature $T = 0°\text{C}$, call this $pI_{T0}^c = 7.2$
  Calculate initial cell value for pI^c at temperature T_C, $pI^c = pI_{T0}^c - 0.016T_C$
  Set initial cell value for $a = -10$
  Calculate initial value for $z_{Hb} = a(pH^c - pI^c)$
  Set initial value for $I77 = 7.216$
  Set initial value for $I78 = 0.4$
  Calculate initial value for factor $pHfact = e^{x^2}$ *where* $x = \frac{pH^c - I77}{I78}$
end procedure

Algorithm 2h Compute RS from code & paper: (part 08 of 14) : membrane0

procedure RS(H)
  [[Set initial value for $\Phi^{HA} = 0$]]
  Calculate initial value for temp dependence factor on pump fluxes $\tau_{Flux}^P = \exp\left(\frac{T_{C0} - T_C}{10}\right)\log\left(Q10^P\right)$
  Calculate initial value for temp dependence factor on passive fluxes $\tau_{Flux}^L = \exp\left(\frac{T_{C0} - T_C}{10}\right)\log\left(Q10^L\right)$
  Calculate initial value for $E = -\left(\frac{RT_K}{F}\right) log\left(r_A\right)$
end procedure

Algorithm 2i Compute the initial RS from code and paper: (part 09 of 14) : cell0

procedure RS(I)
  Calculate initial value for $Q_X = V_w^c(C_{Na}^m + C_K^m + C_A^m + C_B^m + C_Y^m + C_{Mg}^m + C_{Ca}^m) - Q_{Na}^c - Q_K^c - Q_A^c - f_{Hb}Q_{Hb}^c - Q_{Mgfree}^c - Q_{Cafree}^c - Q_{benz2}^c$
  Calculate initial value for $z_X = -\frac{(z_A Q_A^c + z_{Na}Q_{Na}^c + z_K Q_K^c + z_{Mg}Q_{Mg}^c + z_{Hb}Q_{Hb}^c)}{Q_X^c}$
  Calculate initial value for $Q_{(-)}^c = z_{Hb}Q_{Hb}^c + z_X Q_X^c + z_{Mg}Q_{Mg}^c$
end procedure

Algorithm 2j Compute RS from code & paper: (part 10 of 14) : tracers

procedure RS(J)
  Set initial cell value for $I71 = 1$
  Set initial cell value for $Q^c_{NaTracer} = 0$
  Set initial cell value for $Q^c_{KTracer} = 0$
  Set initial cell value for $Q^c_{ATracer} = 0$
  Set initial cell value for $C^c_{NaTracer} = 0$
  Set initial cell value for $C^c_{KTracer} = 0$
  Set initial cell value for $C^c_{ATracer} = 0$
  Set initial cell value for $C^m_{NaTracer} = 1000$
  Set initial cell value for $C^m_{KTracer} = 1000$
  Set initial cell value for $C^m_{ATracer} = 1000$
end procedure

Algorithm 2k Compute RS from code & paper: (part 11 of 14) : NaK Pump

procedure RS(K)
  Set value for constants in Φ^{max}_{Na} equation, $K^c_{m,Na} = 0.2$
  Set value for constants in Φ^{max}_{Na} equation, $K^c_{I,K} = 8.3$
  Set value for constants in Φ^{max}_{Na} equation, $K^m_{m,K} = 0.1$
  Set value for constants in Φ^{max}_{Na} equation, $K^m_{I,Na} = 18.$
  Set initial cell value for $\Phi^{PF}_{Na} = -2.61$
  Set initial cell value for $\Phi^{PR}_{Na} = 0.0015$
  Calculate initial cell value for $\Phi^P_{Na} = \Phi^{PF}_{Na} + \Phi^{PR}_{Na}$
  Calculate initial value for $\Phi^P_K = \Phi^P_{Na}/1.5$
  Calculate initial value for factor $f1 = \left(\frac{C^c_{Na}}{C^c_{Na} + K^c_{m,Na}\left(1 + \frac{C^c_K}{K^c_{I,K}}\right)} \right)^3$
  Calculate initial value for factor $f2 = \left(\frac{C^m_K}{C^m_K + K^m_{m,K}\left(1 + \frac{C^m_{Na}}{K^m_{I,Na}}\right)} \right)^2$
  Calculate initial value for factor $r1 = \left(\frac{C^c_K}{C^c_K + K^c_{I,K}\left(1 + \frac{C^c_{Na}}{K^c_{m,Na}}\right)} \right)^2$
  Calculate initial value for factor $r2 = \left(\frac{C^m_{Na}}{C^m_{Na} + K^m_{I,Na}\left(1 + \frac{C^m_K}{K^m_{m,K}}\right)} \right)^3$
  Calculate initial value for $\Phi^{Fmax}_{Na} = -\frac{\Phi^{PF}_{Na}}{f1 \times f2 \times mgfact \times pHfact}$
  Calculate initial value for $\Phi^{Rmax}_{Na} = -\frac{\Phi^{PR}_{Na}}{r1 \times r2 \times mgfact \times pHfact}$
end procedure

Algorithm 2l Compute RS from code & paper: (part 12 of 14) : NaK2A Cotransport

procedure RS(L)
  Set initial values for temp dep factor for Co flux ...
  Set initial $\kappa^{Co} = 10^{-9}$ *[or* $= 10^{-6}$*]*
  Set initial values for $\Phi^{Co} = 0$
  Calculate initial value for $d^{Co} = \frac{\left(\frac{\Phi^{Co}}{\kappa^{Co}} + (C_A^c)^2 C_{Na}^c C_K^c\right)}{(C_A^m)^2 C_{Na}^m C_K^m}$
  Calculate $\Phi^{Co} = -\kappa^{Co}\left[C_{Na}^c C_K^c (C_A^c)^2 - d^{Co} C_{Na}^m C_K^m (C_A^m)^2\right]$
  for each ion i = Na, K **do**
    Calculate initial values for $\Phi_i^{Co} = \Phi^{Co}$
  end for
  Calculate initial values for $\Phi_A^{Co} = 2 \times \Phi^{Co}$
end procedure

Algorithm 2m Compute RS from code & paper: (part 13 of 14) : Electrodiffusion

procedure RS(M)
  for each ion i = Na, K **do**
    Set initial $\Phi_i = 0$
    Calculate initial value for $\Phi_i^G = f^G\left(\Phi_i - \Phi_i^P - \Phi_i^{Co}\right)$
    Calculate initial value for $\kappa_i^G = -\left(\frac{\Phi_i^G}{z_i \frac{EF}{RT}}\right) \frac{\left(1 - \exp\left[-\frac{z_i EF}{RT}\right]\right)}{\left(C_i^c - C_i^m \exp\left[-\frac{z_i EF}{RT}\right]\right)}$
  end for
  Set initial $\kappa_H^G = 2 \times 10^{-10}$
  Calculate initial value for $\Phi_H^G = -\kappa_H^G \frac{z_H EF}{RT} \frac{\left(C_H^c - C_H^m \exp\left[-\frac{z_H EF}{RT}\right]\right)}{\left(1 - \exp\left[-\frac{z_H EF}{RT}\right]\right)}$
  [[Incidently $\phi = \frac{(c - me^{-k})}{(1-e^{-k})} = paper = \frac{(m - ce^{k})}{(1-e^{k})} = his\ \ code$]]
  Set initial value for $\kappa_A^G = 1.2$
  See below where calculate initial value for Φ_A^G in passive CA fluxes section
end procedure

Algorithm 2n Compute RS from code & paper: (part 14 of 14) : Na:A, K:A Cotransport

procedure RS(N)
  for each ion i = Na, K **do**
    Calculate initial value for $\Phi_i^{CA} = \Phi_i - \Phi_i^P - \Phi_i^{Co} - \Phi_i^G$
    Calculate initial value for $\kappa_i^{CA} = -\frac{\Phi_i^{CA}}{(C_i^c C_A^c - C_i^m C_A^m)}$
  end for
  Calculate initial value for $\Phi_A^{CA} = \Phi_{Na}^{CA} + \Phi_K^{CA}$
  Calculate initial value for $\Phi_A^G = -\Phi_A^{CA}$
  [[Note $\Phi^{HA} = \Phi_H^{HA} = \Phi_A^{HA} = 0$ – to start - in hscreen or membrane]]
  Set initial $\kappa^{HA} = 2.58 \times 10^8$
end procedure

Algorithm 3 Compute RS for the Medium: (part 1 of 1)

1: **procedure** RS(A)
2: Set values for the constants $z_{Mg_F} = 2$
3: Set initial values for the variables $T_C = 37$, buffer = HEPES, $pH^m = 7.40$,
4: Set initial values for the variables $C^m_N a = 145$, $C^m_K = 5$, $C^m_{Y_-} = C^m_{Y_+} = C^m_{Y_0} = 0$, $C^m_{BTotal} = 10$
5: Set initial values for the variables $C^m_{MgTotal} = 0.2$, $C^m_{CaTotal} = 1$
6: Set initial values for the variables $C^m_{NaTracer} = C^m_{KTracer} = C^m_{ATracer} = 1000$
7: Calculate initial value for $T_K = T_C + 273.15$
8: Calculate initial value for $pKa = 7.83 - 1.4 \times 10^{-2} \times T_C$
9: Calculate initial value for $C^m_H = 10^{-pH^m}$
10: Calculate initial value for $K_B = 10^{-pKa}$
11: Calculate initial value for $C^m_{BBound} =$
12: Calculate initial value for $C^m_{BFree} = C^m_{BTotal} - C^m_{BBound}$
13: Calculate initial value for $C^m_{MgFree} =$ [[An iteration is used here]]
14: Calculate initial value for $C^m_{MgBound} = C^m_{MgTotal} - C^m_{MgFree}$
15: Calculate initial value for $C^m_{CaFree} =$ [[An iteration is used here]]
16: Calculate initial value for $C^m_{CaBound} = C^m_{CaTotal} - C^m_{CaFree}$
17: Calculate initial value for $C^m_A =$ not he set it in his code
18: Calculate initial value for $C^m_{Sum+} =$
19: Calculate initial value for $C^m_{Sum-} =$
20: Calculate initial value for $C^m_{Sum} =$
21: Calculate initial value for $C^m_{ENOut} =$
22: **end procedure**

Algorithm 4a Compute RS for the Cell: (part 1 of 2)

procedure RS(A)
    Set values for the constants $z_{Mg_F} = 2$
    Set initial values for the variables $Ht = 1 \times 10^{-6}$, $MCHC = 34$, $v_{lysis} = 1.45$, $b =$, $c =$, $C_N^c a = 10$
    Set initial values for the variables $C_K^c = 140$, $C_A^c = 95$, $Q_{Mg_T}^c = 2.5$, $C_{Mg_F}^c initial = 0.02$
    Set initial values for the variables $Mg_{B_0} =$, $Mg_{B_1} =$, $Mg_{B_2} =$, $KD_{Mg_{B_1}} =$, $KD_{Mg_{B_2}} =$
    Set initial values for the variables $d_{Mg} =$, $y_{Mg} =$, $Q_{Ca_T}^c =$, $C_{Ca_F}^c =$, $Ca_{B_0} =$, $Ca_{B_1} =$, $Q_{benz}^c =$
    Set initial values for the variables $KD_{Ca_{B_1}} =$, $KD_{Ca_{B_2}} =$, $d_{Ca} =$, $y_{CA} =$,
    Set initial values for the variables $a =$, $pI_{T0}^c =$, $pI^c =$,
    Set initial values for the variables $C_{NaTracer}^c = C_{KTracer}^c = C_{ATracer}^c = 0$
    Calculate initial value for $HTrel$
    Calculate initial value for V_w^c
    Calculate initial value for $V_w^c prior$
    Calculate initial value for $V_w^c relative$
    Calculate initial value for $MCHCrelative$
    Calculate initial value for $densityrelative$
    Calculate initial value for Q_{Hb}^c
    Calculate initial value for C_{Hb}^c
    Calculate initial value for f_{Hb}
    Calculate initial value for C_{Osm}^c
    Calculate initial value for Q_{Na}^c
    Calculate initial value for Q_K^c
    Calculate initial value for Q_A^c
    Calculate initial value for $C_{Mg_T}^c$
    Calculate initial value for $C_{Ca_T}^c$
    Calculate initial value for C_{benz}^c
    Calculate initial value for $C_{Mg_F}^c$
    Calculate initial value for $C_{Mg_B}^c$
    Calculate initial value for $Q_{Mg_F}^c$
    Calculate initial value for $Q_{Mg_B}^c$
    Calculate initial value for $MgBT$ Term
    Calculate initial value for $MgATPPT$ Term
    Calculate initial value for $MgDPGPT$ Term
    Set initial values for $C_{Ca_F}^c initial =$
    Calculate initial value for $C_{Ca_F}^c$
    Calculate initial value for $C_{Ca_B}^c$
    Calculate initial value for $Q_{Ca_F}^c$
    Calculate initial value for $Q_{Ca_B}^c$
    Calculate initial value for $AlphaCaTerm$ Term
    Calculate initial value for $CaB0$ Term
    Calculate initial value for $CaB1$ Term
    Calculate initial value for $CaB2$ Term
    Calculate initial MEMBRANE value for r_A
    Calculate initial MEMBRANE value for r_H
end procedure

Algorithm 4b Compute RS for the Cell: (part 2 of 2)

1: **procedure** RS(B)
2: Calculate initial value for C_H^c
3: Calculate initial value for Q_H^c
4: Calculate initial value for $pI^c =$
5: Calculate initial value for $pH^c =$
6: Calculate initial value for z_{Hb}
7: Calculate initial value for $Q_{Hbcharge}^c$
8: Calculate initial value for $C_{Hbcharge}^c$
9: Calculate initial value for C_X^c
10: Calculate initial value for z_X
11: Calculate initial value for $Q_{(-)}^c$
12: Calculate initial value for $C_{(-)}^c$
13: Calculate initial value for C_{sum}^c
14: Calculate initial value for Q_{sum}^c
15: Calculate initial value for C_{ENin}^c
16: Calculate initial value for $Q_{NaTracer}^c$
17: Calculate initial value for $Q_{KTracer}^c$
18: Calculate initial value for $Q_{ATracer}^c$
19: **end procedure**

Algorithm 5 Compute RS for the Membrane: (part 1 of 1)

1: **procedure** RS(A)
2: Set values for the constants ...
3: Set initial values for the variables...
4: Calculate initial value for ...
5: **end procedure**

Chapter 6

Software Design and Implementation

The design decisions made for the software product are described in this chapter.

From the requirements engineering process the software product was decided to be written in the Java programming language for the reasons already mentioned. Some aspects of the Java programming language will be mentioned in the subsequent sections.

6.1 Software Architectural Pattern

Since the system requires the functionality of both user input and the display of recalculated variables, a graphical user interface (GUI) was an obvious choice to effect this. And, moreover, since it was decided to use a GUI with the system the model-view-controller (MVC) system architecture was an obvious choice for the system architectural design pattern. An MVC software architectural pattern was chosen to model the software system.

Following the desired requirement of user input a graphical user interface was chosen as an appropriate means to effect this. Consequently The software architectural pattern adopted for the product was the Model-View Controller pattern.

The Model component consists of several objects (defined by classes) which detail the mathematical model of the system. The model component stores the values of the variables used in the mathematical model. It also provides the methods to calculate the values of the variables using the equations of the mathematical model. In addition it provides methods to retrieve or set the values of all of the variables in the mathematical model.

The View component consists of objects (defined by classes) which display the values of variables in the mathematical model. The display will be by means of a GUI. Some variables are set to initial default values, but their values can be modified by user input, if desired. Such variables will have their values placed in editable JTextFields to facilitate possible user modification. Other variables are calculated using the equations of the mathematical model. They are placed in non-editable JTextFields. These two types of variables are referred to

here as set variables and calculated variables. Set variables are altered by changing the value in an editable JTextField.

The Controller component consists of parts of the same objects as the View objects. Thus the Controller components are contained with the related View component class. A controller event is triggered when a JButton that is listened to by the ActionEvent Listener is pressed. One type of button is to recalculate the values of the calculated variables using the equations of the mathematical model and redisplay the new values in the GUI. Another types of button is to accept the values of all of the set and calculated variables and move on to the next stage. Moreover the Controller component also prints the results of the calculations to output files.

To reiterate the view and controller components were contained in a combined View-Controller objects/class.

6.2 General Considerations

To clarify the later discussion some aspects of object and classes are now discussed.

A software object characterises the state and behaviour of a distinct aspect of the system. The state is described by values in variables. Note that variables in the computer science sense are used for symbolic labels used to describe variables, parameters, and constants in the scientific sense. The behaviour of the object is characterised by its methods which can alter the state, i.e., the variables in the object.

An important principle of object-oriented programming is that the details of the variables and the methods should, in general be hidden from a user of the software. This is referred to as data encapsulation. However, variables can be retrieved and modified if the software is created to use safe get/set methods.

Objects are defined by a class. Specifically, an object is created as an instance of a class. The class enables different versions, or instances, of the object to be created, in principle. The class describes all of the details needed to create an object.

Using objects is helpful because: it forces the code to be written in a modular manner. This means different logical parts can be written and maintained independently, in principle. The object with its state and behaviour can be created and passed around inside the "system".

Thus to create the software we will start be describing the classes that were used to define the system.

6.3 Classes Design

Before discusses class design some mention should also be made about the states of the system. The software was to be designed to follow some aspects of the general software design of the current old program. Specifically, the new code followed the design structure

of different states of the system. That is, there was an initial state, called the reference state. When an simulated experiment was run this state was succeeded by the dynamic state. During the dynamic state, a step is taken where a small perturbation is made to some variables. Then these in turn change other variables. The new values for variables are determined at each step. Then the system goes on to the next step, and so on, until the experiment finishes.

I follow the separation of concerns concept where I divide the system into logical functionally separate parts. In this model the parts are not functionally independent but are coupled, since changes in the values of variable in one part affect variables in another part.

The manner in which the old code is written is that some calculations are handled with subroutines, some are carried out in sequence. The variables and equations are not organised clearly separated areas. However, the code is arranged partly into related sections where user input is requested. The sections are arranged according the physical processes, type of ion etc.

In designing the software it is important to organise it in a logical manner. After reflection it was decided that a logical organisation was to store variables and the methods to set, get and calculate them according to the physical region in which they occurred.

That is, the system was envisaged to consist of a single red-blood cell used to model red-blood cells collectively, the extra-cellular fluid which is referred to as the medium. These regions are represented by a Cell class and a Medium class, Since these regions/classes will share some characteristics, for example, both will have their own concentrations of Na^+ ions, it seemed sensible to define a Region superclass which defined most characteristics of both regions. These characteristics include the variables in those regions and the methods to set, get, and calculate them. The Cell and Medium classes extend the Region class. That is, the Cell and Medium classes are subclasses of the Region superclass.

Since in the real world the transportation of substances between the cell and the medium occur across the membrane due to a number of different processes, it seemed sensible to model cell transport processes via a Membrane class. This class contains the models describing transportation processes and the variables affected by those processes.

The above classes are Model classes which store variables with methods to calculate them defined according to the mathematical model of RBCs, as well as get and set methods.

The View-Controller classes handle the state of the system at different stages. For example, the initial state of the system, called the reference state, is set up with the ReferenceState class. This calls the Model classes and displays the values of variables on a GUI. It also facilitates user input and the recalculation of some variables. This class calls the ReferenceReport class to print the values of all variables to an output file.

After the Reference State, the View-Controller classes DynamicState takes over. It facilitates user input regarding the condition of the proposed simulated experiment and runs the experiment based on the input or default experimental conditions and recalculates the variables calling methods in the model classes.

6.4 Model Classes/Objects.

The Model classes which are used to create model objects are the following. I describe each.

Region class This class defines a superclass for the two classes defined by physical regions, namely the Cell and Medium classes. The Region class is used to store the variables and methods relevant to those regions. Apart from various parameters, the main variables are the concentration of substances.

Cell class This class extends the Region class to store variables and methods relevant to the red-blood cell.

Medium class This class extends the Region class to store variables and methods relevant to the extra-cellular fluid, or medium.

Membrane This class is used to store the values of (principally) the fluxes of substances into and out of the cell and the models used to calculate, set and get those variables. It also stores variables dependent on both Cell and Medium variables.

Experiment This class is used to store the variables to define the simulated experimental conditions. This class calls all of the above methods to determine values during the experiment.

6.5 View-Controller Classes

ReferenceState class This class is used to set up the initial experimental conditions called the reference state. It calls the Cell, Medium, and Membrane classes. It enables the initial state to be modified via user input and recalculation using the aforementioned model classes.

ReferenceReport class This class prints out the values of all variables for the reference state.

DynamicState class This class is used to run the simulated experiment to determine the values of variable at each step in the experiment. It calls the Cell, Medium, Membrane, and Experiment classes. It enables additional options to be included in th experiment before running the experiment.

DynamicReport class This prints out the values of all variables for the during each step during the dynamic state of the running experiment.

6.6 The Algorithms used to calculate the Reference State

Here I will include the algorithms actually implemented in the new program. The order and structure will differ because of my different classes.

6.7 The Algorithms used to calculate the Dynamic State

Here I will include the algorithms actually implemented in the new program. The order and structure will differ because of my different classes.

6.8 Creation of the Code

The algorithms mentioned previously were used to write the new code.

Note I did not incorporate the EDTA option. Nonetheless the software can be tested for the stage it is at to see if what is written so far works as expected,

Chapter 7

Software Evaluation

As mentioned earlier, the software product has not yet incorporated the EDGTA options (i.e., the screen 2 options in the old code). I believe other options are incorporated in the code, beven if they are not yet at a stage where they can easily changed by user input.

Nonetheless other functionality of the program can be tested. It should be noted that the only tests performed were to do with correctness testing. That is, the testing was performed to verify validate the correctness of the implementation in regards to the calculations and algorithms. To do this testing the results of running the old code for a set of preliminary test cases were carried out and compared with the results running the new code.

Since the GUI is not wholly functional preliminary tests were carried out running the default values. In these series of test, the original old code was run on an older laptop computer running Windows XP and the new code was run on a more modern computer using Windows 7. The values of the variables produced by the new code were compared with that for the old code.

A test was carried out to confirm the values of variables in the reference state. This test was passed.

A test was carried out to determine the values of variables after the smallest possible initial integration step. The duration of the experiment in that case was 0.0336 minutes. This test was passed.

A test was carried out running the experiment for a longer duration and printing out the results at the end of each integration. This test was done for 5 minutes. There were several steps performed during that experiment. The values of the variables at each step were checked.

The checking in all of these cases was done by eye, comparing the numbers in printed output files.

Chapter 8

Conclusions

The main aims of this project were to create software which would act as a replacement for the old code and to document the revised RBC model. Each of these was only achieved partially.

The major difficulty with this work was to decipher the old code and convert it into the Java programming language, but organised in an appropriate object-oriented manner. Considerable time and effort was spent doing this. It took much longer than anticipated.

Furthermore, the software/model was created in stages. The first stage imitated the steps to create the reference state in the original paper by Lew and Bookchin (1986). The second stage tried to imitate the steps in the old code to create the reference state. This was created using a GUI interface which functioned well. The third stage was to create the new code using the object-oriented design mentioned in Chapter 6. This was done but the a fully functioning GUI was not achieved due to the added complexity of the code compared to the simpler stages considered earlier.

Nonetheless the code could be run with default values and the variables relating to the duration of the experiment and the number of integration before printing could be editted in the code. This was how the preliminary tests were carried out. These seem to show that the code is working as expected. Otherwise small changes would be amplified and become very obvious. This indeed happened during the debugging stage. To sort such problems the original source code was edited with PRINT statements and numerous variables were checked.

Chapter 9

Future Work

The code could be improved by fully incorporating the GUI interface with action listeners for the setting of the reference state. A functioning GUI is also required for the Dynamic State to enable user input and modifications of the simulated experiments.

The documentation of the revised RBC model could be completed. Ideally a glossary should be created to make modifications of the code easier for a programmer. In addition a user manual should be created for the software to make the software easier to use.

Once these are working a GUI could possibly be created to including the plotting of the results onto a GUI display as the experiment runs or after the experiment finishes.

Bibliography

J. C. Freedman and J. F. Hoffman. Ionic and osmotic equilibria of red blood cells traced with nystatin. *J. Gen. (Physiol.*, 74:157–185, 1979.

V. L. Lew. Concise guide to the red cell model programme. *Draft manuscript*, 2000. 18 pages.

V. L. Lew. Mg-dependence of the ca pump: Extension to include passive & active ca^{2+} & mg^{2+} transport and cytoplasmic ca^{2+} & mg^{2+} buffering. *Draft manuscript*, 2014a. 10 pages.

V. L. Lew. M1.bas. *Draft copy of BASIC programme*, 2014b. 62 pages.

V. L. Lew. Modelling cellular homeostasis: General principles and practical rules for constructing useful models. *Draft manuscript*, 2014c. 10 pages.

V. L. Lew and R. M. Bookchin. Ion transport pathology in the mechanism of sickle cell dehydration. *J. Membrane Biol.*, 92:57–74, January 1986.

V. L. Lew, C. J. Freeman, O. E. Ortiz, and R. M. Bookchin. A mathematical model of the volume, ph, and ion content regulation in recticulocytes: Application to the pathophysiology of sickle cell dehydration. *Jl. Clin. Invest.*, 87:100–112, January 1991.

P. D. McConaghy and M. Maizels. The osmotic coefficients of haemoglobin in red cells under varying conditions. *J. Physiol. (London)*, 155:28–45, 1961.

T. F. Weiss and V. L. Lew. Cellular homeostasis simulator. *Draft manuscript*, 2014. 126 pages.