Modelling Mitochondrial Complex IV Bioenergetics

by

Chris Cadonic

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfilment of the requirements of the degree of

MASTER OF SCIENCE

Graduate Program in Biomedical Engineering
University of Manitoba
Winnipeg

UNIVERSITY OF MANITOBA

Abstract

Faculties of Health Sciences, Science and Engineering
University of Manitoba

MASTER OF SCIENCE

Modelling Mitochondrial Complex IV Bioenergetics

by Chris CADONIC

A computational model for mitochondrial function has been developed from oxygen concentration data measured in the Oroboros Oxygraph-2k and oxygen consumption rates measured in the Seahorse XF24 Analyzer. Measurements were acquired using embryonic-cultured cortical neurons and isolated mitochondria from CD1 mice. Based on the biological mechanism of mitochondrial activity, a computational model was developed using biochemical kinetic modelling. To modulate mitochondrial activity, dysfunctions were introduced by injecting the inhibiting reagents *oligomycin*, *rotenone*, and *antimycin A*, and the uncoupling reagent *carbonyl cyanide 4-(trifluoromethoxy)phenylhydrazone* (*FCCP*) during measurements. To incorporate these changes, model equations were adapted and globally calibrated to experimental data using the genetic algorithm developed by Jason Fiege of the University of Manitoba by fitting oxygen concentration data. The model was coded in MATLAB R2014a along with the development of a graphical user interface for simulating mitochondrial bioenergetics *in silico*.

Acknowledgements

Project Funding:

I thank my main funding sources for finacially assisting my efforts throughout this project: the *UMGF* granted by the University of Manitoba and the *Alexander Graham Bell CGS-M* granted by *NSERC*. I also thank the *Scottish Rite Charitable Foundation of Canada*, the *Alzheimer's Society of Manitoba*, *FGS* at the University of Manitoba, and *Seahorse Biosciences* for the awards and travel funding provided to me during this project. I also thank the *NSERC* grants provided to my supervisors Dr. Benedict Albensi and Stephanie Portet, for also supplying some assistance in purchasing supplies and additional funding for my project.

Personal thanks:

I wholeheartedly thank Drs. Benedict Albensi and Stephanie Portet for their devotion and assistance in guiding me through this project. I thank Dr. Wanda Snow, Dr. Jelena Djordjevic, Danielle McAllister, and Danial Peirson for their support and assistance in the lab. I thank Dr. Jason Fiege for his gracious assistance with optimization, and for setting me up on his optimizer suite *Qubist* and specifically on the genetic algorithm tool *Ferret*. I thank Ella Thomson for her assistance in the laboratory, and for all the work that she did in the lab. I thank Dr. Francis Amara, Dr. Douglas Thomson, and Dr. Jason Treberg for their input, support and interest to serve on my masters committee. I thank my family for supporting me throughout my academic career, and my brother for assisting with some of my figures. And last but certainly not least, I thank Samantha Phrakonkham for her support, love, and most of all for keeping me sane all throughout my masters project.

Dedicated to my family and my loving girlfriend Samantha

Contents

List of Tables

List of Figures

Abbreviations

ATP	**A**denosine **Tri**p**ph**osphate
ADP	**A**denosine **Di**p**ph**osphate
Cyt c_{ox}	**Cyt**ochrome **c** in unreduced state (**ox**idized relative to reduced form)
Cyt c_{red}	**Cyt**ochrome **c** in **red**uced state
AD	**A**lzheimer's **D**isease
ETC	**E**lectron **T**ransport **C**hain
FBS	**F**etal **B**ovine **S**erum
FCCP	**C**arbonyl **c**yanide 4-(tri**f**luoromethoxy)**p**henylhydrazone
DMEM	**D**ulbecco's **M**odified **E**agle **M**edium
GA	**G**enetic **A**lgorithm
GUI	**G**raphical **U**ser **I**nterface
HBSS	**H**ank's **B**alanced **S**alt **S**olution
HBSS-S	**H**ank's **B**alanced **S**alt **S**olution **S**upplemented
HEPES	4-(2-**h**hydroxy**e**thyl)-1-**p**iperazine**e**thane**s**ulfonic acid
IMS	**I**nter**m**embrane **s**pace
MSE	**M**ean-**s**quared **E**rror
NAD	**N**icotin**a**mide **D**inucleotide
NADH	**N**icotin**a**mide **D**inucleotide with **H**ydrogen (reduced form)
NB	**N**euro**b**asal growth medium
NB-S	**N**euro**b**asal growth medium **S**upplemented
NF	**N**euro**F**ilament Buffer
N side	Mitochondrial matrix (**N**egative side)

$\boldsymbol{o(t)}$	Oxygen concentration as a function of time
OCR	**O**xygen **C**onsumption **R**ate
pmf	**p**roton-**m**otive **f**orce
P side	Intermembrane space (**P**ositive side)
$\mathbf{P}_i$	**I**norganic **P**hosphate
Q	Ubi**q**uinone
$\mathbf{QH}_2$	Ubi**q**uinol (reduced form)
$\boldsymbol{r(t)}$	Cytochrome c reduced concentration as a function of time
$\boldsymbol{\rho(t)}$	Intermembrane space proton concentration as a function of time
$\boldsymbol{\omega(t)}$	Matrix proton concentration as a function of time

Chapter 1

Introduction

1.1 Background

Cellular mitochondria are the primary source for biological energy production in the form of adenosine triphosphate (ATP) [1][2]. Considering the inherent complexity and variability of biological systems, there are numerous alterations in the properties of the mitochondria that may be found in certain cell types, but not others. For instance, increased levels of mitochondrial cristae folding are found in liver cells compared to many other cell types [3]. Additionally, the mitochondria also show variations in structure and/or function due to genetic mutations or disease [2][4].

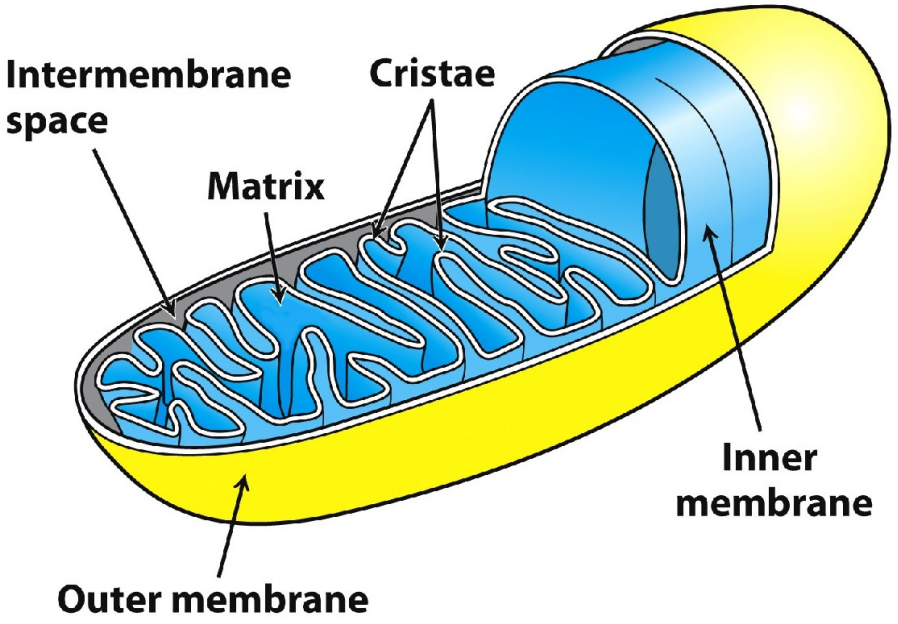
To adequately describe the properties of mitochondria that may be altered in these states, the standard properties of mitochondrial structure and function must first be firmly established.

1.1.1 General Mitochondria Biochemistry

The general characteristics of the mitochondrion, such as structure and general function, have been explained in great depth [1][2]. The general structure of the mitochondria is

shown in Fig. 1.1.

Figure 1.1: A protoytypical mitochondrion. A graphical representation of a typical mitochondrion. Each mitochondrion contains two separate membranes, both inner and outer membranes. The innermost space, enclosed by the inner membrane, is known as the mitochondrial matrix. The space enclosed by the inner and outer membranes is known as the intermembrane space. Folds in the inner membrane, known as cristae, allow for increased surface area of the inner membrane, where the bioenergetic machinery of the mitochondria are housed. Figure modified from [5].



The mitochondria contain an outer membrane that is permeable to small molecules and ions. The unique nature of the mitochondria begins with the existence of an inner membrane, that is impermeable to most molecules and ions. This membrane, referred to as the *inner mitochondrial membrane*, separates the mitochondria into two major subspaces: an inner space referred to as the *mitochondrial matrix*, and a small region between membranes referred to as the *intermembrane space* (IMS). The most important role of this inner mitochondrial membrane, however, is that it houses a system of protein complexes known as the *electron transport chain* (ETC) [1][3][6].

The role of the ETC is to use potential energy stored in oxidizable compounds, such as glucose, to be converted into the form of energy used at the cellular level, which is ATP. The ETC requires the transport and processing of these oxidizable compounds

into the mitochondrial matrix. The result of this processing is the transfer of electrons from oxidizable compounds to the ETC primarily by using carrier molecules known as *Nicotinamide dinucleotide (NADH)*. Once they are delivered to the ETC, they can be carried by the protein complexes in this system to generate ATP.

The ETC consists of a chain of reducing agents organized into four major protein-metal complexes:

- NADH-coenzyme Q oxidoreductase, or *complex I*,
- succinate-coenzyme Q oxidoreductase, or *complex II*,
- Coenzyme Q-cytochrome c oxidoreductase, or *complex III*,
- cytochrome c oxidase, or *complex IV*.

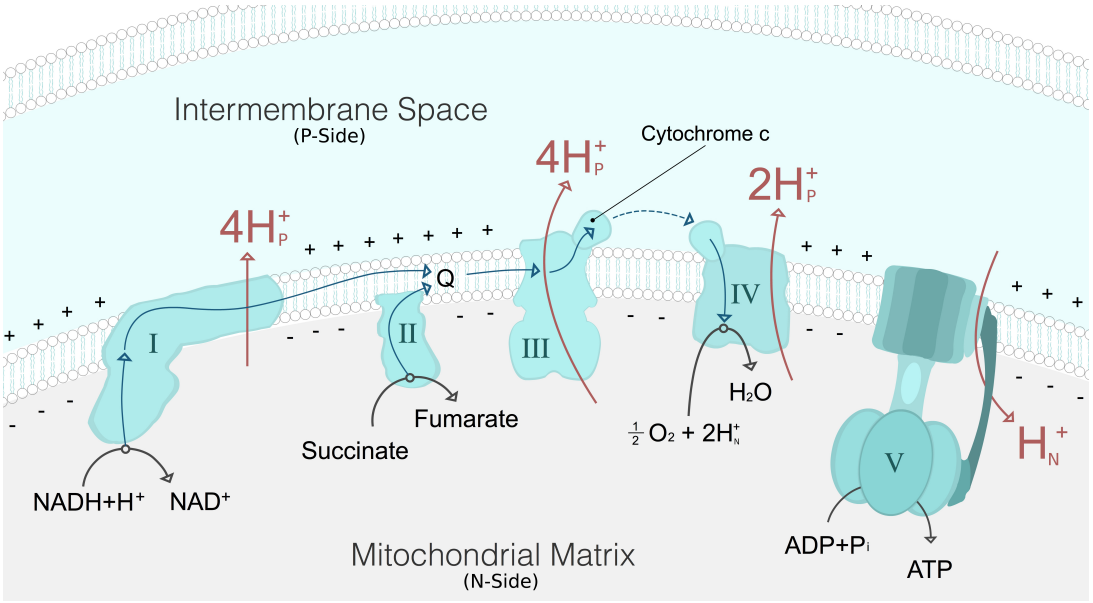
An additional integral protein that is closely related to the function of the ETC is *ATP synthase*, also referred to as *complex V*, which carries out ATP synthesis as a result of electron flow through the ETC. Complex V is found primarily in the folds of the cristae in the inner mitochondrial membrane, adjacent to ETC systems [7].

Electrons delivered to the ETC propagate through the system of proteins from input at complexes I and II, up until the final electron acceptor, molecular oxygen, O_2, in complex IV. The reactions that allow for electron propagation through these complexes are detailed in the following section.

1.1.2 Biochemical Reactions in the Mitochondria

As electrons are passed through each complex, as shown in Fig. 1.2, they are sequentially transferred from one redox centre to another. This series of electron transfers generates a small amount of energy that is effectively used to transport protons (H^+) across inner mitochondrial membrane through complexes I, III and IV in the ETC. This transport

Figure 1.2: The Electron Transport Chain. This figure illustrates the electron transport chain (ETC) of the mitochondria, along with ATP synthase or complex V. There are four primary protein complexes in the electron transport chain, complexes I-IV, and an ancillary protein complex that accompanies ETC sites, complex V. Electron flow, shown by the blue arrow, starts in the ETC at complexes I and II, and ends as electrons are taken up by molecular O_2. Red arrows show the translocation of protons across the inner mitochondrial membrane through complexes I, III, IV, and V. More detail about the function of the ETC is detailed in Sections 1.1.2 and 1.1.3. Figure developed with assistance from: Alvin Cadonic.



thereby creates an electrochemical gradient across the membrane that favours H^+ flow back through complex V [1].

This electrochemical gradient, known as a *protonmotive force* or *pmf*, represents a potential that drives the flux of protons back through complex V in towards the matrix [1]. This *pmf* has two components, the chemical potential energy due to a difference in H^+ concentration, and the electrical potential energy due to the charge differential when protons are separated across an impermeable membrane. Since charge associated with proton build-up on the outside of the membrane creates a net positive charge, the intermembrane space is referred to as the positive or *P* side, whereas the mitochondrial matrix becomes the negative or *N* side [1]. Protons located in the P side are designated H^+_P, whereas protons located in the N side are designated H^+_N.

The overall reaction for generating this *pmf* is [1]:

$$NADH + H^+ + \frac{1}{2}O_2 + 10H_N^+ \longrightarrow NAD^+ + H_2O + 10H_P^+. \tag{1.1}$$

This equation shows that a single NADH delivering two electrons reduces a half mole equivalent of O_2, producing a single molecule of H_2O and pumping ten protons across the membrane. The activity of each complex is discussed separately below in relation to the delivery of a single set of electrons from one NADH molecule.

In the overall reaction given by equation (1.1), this does not make clear how each complex is involved in transferring electrons from the original delivery agent NADH to O_2. As NADH are delivered to the ETC, complex I binds to NADH and catalyzes the redox reaction between NADH and *ubiquinone* (Q) [2]. This results in the transfer of electrons from NADH to ubiquinone, forming *ubiquinol* (QH_2). This reaction is coupled to the translocation of protons from the mitochondrial matrix to the intermembrane space, and has an overall reaction of [1]:

$$NADH + H^+ + 4H_N^+ + Q \longrightarrow NAD^+ + QH_2 + 4H_P^+ \tag{1.2}$$

Electrons can also enter the ETC through complex II, where electrons are shuttled using the carrier $FADH_2$ instead of NADH, following the reaction:

$$Succinate + 2H_N^+ + FAD \rightleftharpoons Fumarate + FADH_2 \tag{1.3}$$

Electrons in $FADH_2$ are quickly then shuttled through complex II and are once again transferred to ubiquinone to form QH_2[1]:

$$FADH_2 + Q \rightleftharpoons FAD + QH_2 \tag{1.4}$$

yielding an overall reaction of:

$$Succinate + 2H_N^+ + Q \rightleftharpoons Fumarate + QH_2 \tag{1.5}$$

In complex I, four protons are translocated into the P side, while complex II results in no net translocation. Both complex I and complex II act as entry points for electrons into the ETC, resulting in the production of QH_2 to carry electrons toward complex III. The QH_2 transfers electrons to complex III, which then flow to the next electron acceptor: cytochrome c ($cyt\ c_{ox}$)[1][2]. In the naturally occurring state of cytochrome c, it is considered oxidized relative to its reduced form following electron acquisition. Once cytochrome c accepts electrons, it is converted from the $cyt\ c_{ox}$ form to $cyt\ c_{red}$. QH_2 effectively reduces two molecules of cytochrome c, transferring a single electron to each [1]:

$$QH_2 + 2cyt\ c_{ox} + 2H_N^+ \longrightarrow Q + 2cyt\ c_{red} + 4H_P^+ \tag{1.6}$$

Protons are once again taken up from the N side and then shuttled through complex III into the P side.

Electrons can now enter complex IV by the activity of cytochrome c, where they are then transferred to the ETC's final electron acceptor: oxygen. The overall reaction for complex IV is [1]:

$$2cyt\ c_{red} + 4H_N^+ + \frac{1}{2}O_2 \longrightarrow 2cyt\ c_{ox} + H_2O + 2H_P^+ \tag{1.7}$$

In this reaction four reduced cytochrome c's are required for the operation of complex IV, although this discussion involves electron flow from the supply of a single pair of electrons.

Electrons flowing through the complexes pass along various enzyme elements, originating from the carriers NADH and $FADH_2$ and terminating at O_2 [2]. The result of directing

electrons through this system is to generate a *pmf*, which is accomplished by the pumping of protons into the intermembrane space [1][2]. Protons are driven by this *pmf* to translocate back into the mitochondrial matrix through a proton channel located within complex V, which results in the production of ATP.

1.1.3 The Chemiosmotic Theory of Bioenergetics

The *chemiosmotic model* describes the coupling of proton flow with the activity of complex V [8]. The *pmf* generated by proton flow represents both a concentration and electrical gradient across the inner mitochondrial membrane, which directs protons toward the mitochondrial matrix [2]. The chemiosmotic model of ATP production states that proton flow through complex V results in a structural rotation of a set of subunits within the complex, thus creating torque from the flow of protons. This rotational energy is then harnessed to drive the production of ATP through an effective binding of ADP and an inorganic phosphate P_i together, thereby acting to couple proton flow to ATP generation. As this process requires the acceptance of electrons by molecular oxygen O_2, electron flow to oxygen resulting in the production of ATP generated by proton flow is known as *oxidative phosphorylation* [1][2][8].

There are several chemical agents that may affect either the structure or functioning of the ETC, thereby affecting the process of oxidative phosphorylation. The specific agents used in this project are *oligomycin*, *carbonylcyanide-p-trifluoromethoxyphenylhydrazone* (*FCCP*), *rotenone*, and *antimycin a*.

Oligomycin inhibits the activity of complex V by binding to the proton channel segment, disallowing protons to flow through the complex [9]. Since protons can no longer flow through this channel, ATP production is ceased as no rotational energy is generated to produce ATP. The second reagent injected is *FCCP*, which effectively uncouples complex V from the ETC. *FCCP* results in the production of ionophores in the inner mitochondrial membrane, which allows protons to freely flow through the membrane back into the

mitochondrial matrix [1][10]. This decouples complex V from the ETC since the ETC can now operate without using complex V as a proton channel. This also represents a state of maximal activity for the mitochondria, since the ETC can process electrons fully without the resistance generated by the *pmf*. In this state, protons can immediately equilibrate through ionophores created by *FCCP*. The reagents *rotenone* and *antimycin A* result in full inhibition of the system. *Rotenone* inhibits the activity of complex I by blocking the NADH binding site [11], whereas *antimycin A* inhibits complex III by blocking the binding sites for Q [12]. Thus, electrons cannot be effectively delivered to the ETC and cannot produce any QH_2 for complex III, disallowing the ETC to proceed. In this case, oxygen is no longer consumed since the ETC does not accept or transmit a flow of electrons toward O_2.

1.1.4 Objective

Development of a mathematical model to describe the function of a mitochondrion allows for a method to build a simplified representation of an otherwise complex system. Thus, the goal of this project was to develop a mathematical model that could not only describe the functioning of the mitochondria, but to also allow for prediction of mitochondrial functioning under different biological conditions.

The strategy of the modelling approach was to mathematically describe the biochemical reactions detailed above in Section 1.1.2 as kinetic enzymes. Experimental manipulations described in Section 1.1.3 were also incorporated into the model to ensure reproducibility of experimental data. The aim for this model was to build a simulation package that is user-friendly, powerful, and affords description and prediction of mitochondrial functioning *in silico*.

1.2 Modelling the Mitochondria

Models describing the functioning of mitochondria have been developed in many different biological systems, such as in cardiac systems [13][14][15][16][17] and skeletal systems [18]. It is rare, however, to find models that aim to explain mitochondrial functioning specifically in the nervous system. To allow application of a mitochondrial model to neural functioning or neurodegenerative disorders, the limited selection of developed models is problematic since mitochondrial properties are known to show differences in structure and function between different biological systems [19].

Many previous models describe the ETC and its associated reactions using kinetic modelling schemes [13][14][15][16][17][18][20][21], similar to the approach used in the current work. A key difference between previous models, however, regards what components are additionally described by the model. In addition to modelling the ETC using kinetic models and thermodynamic principles, Beard [14] also described phosphate system control (*i.e.* the conversion and translocation of ATP and its derivatives) and transport system fluxes for protons and potassium. Wu et al. [13] expanded this model to focus on the reactions of the citric acid cycle, and relevant citric acid cycle substrate transporters. Korzeniewski and Zoladz [18], and Korzeniewski et al. [17] modelled oxygen consumption rate and proton translocation, as in the current model, but also described the creatine kinase system, proton production/consumption, and mitochondrial/cytosolic volumes.

More complex models of mitochondrial functioning have also been developed, such as the two-compartment redox model of Kembro et al. [22], the thermodynamic redox models of Wei et al. [16] and Cortassa et al. [15]. The model developed by Cortassa et al. involves modelling not only the kinetic properties of the mitochondria, but also the thermodynamic activity of the ETC, the citric acid cycle, and calcium dynamics. Many elements of this model were adapted from and developed from a mathematical model of oxidative phosphorylation developed by Magnus and Keizer [23][24][25] and models of the citric acid cycle developed by Dudycha and Jafri [21]. Further developments of this

model involve modelling additional components such as pH and ion flux rates [16], or introducing regulation of reactive oxygen species networks [22].

Bertram et al. [26] developed a simplified model based on the Cortassa et al. model [15] by reducing the complexity of the underlying flux rates. The motivation for adapting this general model was the completeness and liberal consideration for model elements during derivation.

The approach undertaken in the current project has numerous deviations from the approach taken in previously developed models. First of all, the current model focuses on complex IV activity, due to the nature of the experimental data acquired in this project, as detailed in Section 2.2.1. Secondly, experimental data was acquired from brain tissue, not from cardiac or skeletal tissue. Additionally, while previous models often used simple Michaelis-Menten kinetics, the current model additionally incorporated activation kinetics, outlined in Appendix A.1.4. Furthermore, the current model was developed to reproduce experimental data that detailed both normal and altered functioning of the mitochondria. This was incorporated by adapting the model to represent the mode of action of the chemical reagents used in the experimental protocol, detailed in Sections 1.1.3 and 2.1.2. Mathematical adaptations to the model are outlined in Section 2.2.6. Finally, a graphical user interface (GUI; see Fig. 3.14) was created for user-friendly operation and manipulation of the model. This allows users to simulate mitochondrial functioning under different conditions by changing the components of the model without adapting the model's code.

Chapter 2

Methods

2.1 Biological Methods

Calibration of the mathematical model developed in this project was carried out by optimizing the model to fit experimental oxygen concentration data. This data was acquired by measuring oxygen concentration over time in isolated chambers in the Oroboros Oxygraph-2k using homogenated cortical tissue from CD1 mice. Qualitative behavior of the rate of oxygen consumption rate was validated by measurement in a 24-well plate in the Seahorse XF24 analyzer using cultured cortical neurons from CD1 mice.

2.1.1 Embryonic Cell Cultures

Cultures were carried out on 15-day old timed-pregnant CD1 mice. On the day prior to culturing, specialized Seahorse 24-well plates were coated with poly-D-lysine (Life Technologies) overnight.

On the day of the cell culture, two solutions were prepared. Firstly, the supplemented Neurobasal growth medium (NB-S) was prepared by adding reagents to glutamine-free Neurobasal growth medium (NB; Life Technologies). Reagents added were: 3 mL of 200

mM L-glutamine (Thermo Scientific) per 500 mL of NB, 2.5 mL of 1M HEPES (Thermo Scientific) per 500 mL of NB, 10 mL of B27 Supplement (Life Technologies) per 500 mL of NB, and 5 mL of 10 mg/mL Streptomycin (Calbiochem) per 500 mL of NB. Secondly, supplemented Hank's balanced salt solution (HBSS-S) was prepared by adding 500 mL of 1M HEPES and 5 mL of 10 mg/mL Streptomycin to 500 mL of Hank's balanced salt solution (HBSS; Thermo Scientific). Additionally, on this day the plate was rinsed of poly-D-lysine using double-distilled H_2O and left to air dry for at least 1 hour prior to loading the plate with a cell suspension. Mice were anesthetized and decapitated to euthanize the mother. The embryos were then removed from the mother and placed in cold 70% ethanol for two minutes. Following ethanol treatment, the embryos were then quickly transferred to cold HBSS-S. Embryos were dissected in HBSS-S in a petri dish, seated on ice, using a dissection microscope. Both hemispheres of cortex were removed from each embryo, requiring approximately six embryos for each Seahorse plate. Cortical tissue was suspended in 15 mL of cold HBSS-S. After this time period, 12 mL of HBSS-S was removed, and then warmed NB-S with 5% fetal bovine serum (FBS) was added to top-up the solution to 10 mL. Cortical tissue in this solution was then triturated using a 20 μL pipette, gently breaking up the tissue until the solution became cloudy. This solution was then filtered using a 40 μm cell strainer (BD Falcon) to separate cells and cellular debris. The filtered solution was then topped up to 10 mL using NB-S with 5% FBS. 20 μL of this cell suspension was added to 20 μL of Trypan blue dye (Thermo Scientific) to count out the amount of cells per mL of suspension using a haemocytometer (Hausser Scientific). 400 μL of the cell suspension was then loaded into each non-control well in the Seahorse plate with a density of 300,000 cells per 400 μL. Control wells (wells A1, B4, C3, and D6) were loaded with 400 μL of NB-S with 5% FBS without cells. After loading, the seahorse plate was then incubated at 37°C in 5% CO_2 in air overnight. The following day, 400μL of NB-S with 5% FBS from each well was replaced by 400 μL of NB-S without FBS. At the end of this day, 50 μL of 18 μM *cytosine arabinofuranoside* was added to each well. 24 hours following addition of CA, all 450 μL of media was removed from each well and replaced with 500 μL of freshly made NB-S without FBS.

The plate was left to incubate at 37°C with 5% CO_2 for the remainder of the culture, with 50% media replacements (replacement of 250 μL growth media with freshly made NB-S without FBS) every 3 days.

2.1.2 Seahorse Measurements

Cortical neurons were cultured for 9 days prior to bioenergetic measurements in the XF24 Analyzer (Seahorse Bioscience). The day prior to measuring oxygen concentration levels in the specialized cell plate, a calibration plate was loaded with 1 mL of calibration media (Seahorse Bioscience) in each well and incubated at 37°C overnight. The Seahorse experimental protocol was prepared in the software accompanying the XF24 Analyzer, specifying the reagents to be added and the layout of the cells on the culture plate.

On the day of the experiment, assay media was prepared by adding 600 μL of 100 mM sodium pyruvate (Life Technologies) and 600 μL of 1M glucose in Dulbecco's modified eagle medium (DMEM; Life Technologies) to 59 mL of DMEM. The assay media was then incubated in a water bath at 37°C for 15 minutes. The cell culture plate was removed from the CO_2 incubator and placed in the biosafety cabinet. Then 400 μL of culture media was removed from each well. Immediately after removing culture media, 1 mL of warmed assay media was added to each well. Approximately 1 mL of media was then removed from each well, leaving 100 μL of media in each well. After this wash with warmed assay media, each well was then topped up to 675 μL by adding 575 μL of additional assay media. The Seahorse plate was then incubated at 37°C for 1 hour. During this time, the injection reagents were prepared by: adding 20 μL of 10 mM *oligomycin* to 1980 μL of warmed assay media; adding 20 μL of 10 mM *rotenone* and 20 μL of 10 mM *antimycin A* to 1960 μL of warmed assay media; adding 10 μL of 5 mM *FCCP* to 990 μL of warmed assay media, and then adding 40 μL of this dilution to 1960 muL of warmed assay media. The drugs were then loaded into the calibration loading plate with: 75 μL of *oligomycin* loaded into port A of each non-control well, 83 μL of *FCCP* into port B of each non-control well, 93 μL of the *rotenone/antimycin A* solution into port C of

each non-control well. For control wells A1, B4, C3 and D6, assay media was added to each port with 75 μL in port A, 83 μL in port B, and 93 μL in port C. The calibration plate and calibration loading plate were then loaded into the Seahorse and the protocol was initiated in the software. After 30 minutes of calibration, the calibration plate was ejected and replaced with the Seahorse culture plate. The assay proceeded to inject each reagent at regular intervals and measure oxygen concentration throughout the remaining 90 minutes of analysis. Once complete, the non-control wells of the Seahorse culture plate were then scraped using neurofilament buffer (NF) and stored at -4°C for protein normalization.

2.1.3 Oxygraph Measurements

Homogenate used in the Orobos Oxygraph o2k measurements was prepared from CD1 mouse cortical brain tissue. The white matter was removed from the cortical tissue. The cortical tisse was mechanically homogenized using 500 μL of mitochondrial isolation buffer, and then centrifuged at 800 G for 10 minutes at 4^o Celsius. The supernatant was then centrifuged at 8000 G for 15 minutes at 4^o Celsius. The pellet was resuspended and then centrifuged again at 8000 G for 15 minutes at 4^o Celsius. The pellet, containing isolated mitochondria, was resuspended in mitochondrial isolation buffer. A Bradford protein assay was performed to determine the protein concentration of the homogenate solution.

The oxygraph was calibrated using 2 mL of Budapest buffer at 37^o Celsius. 200 μg of cortical mitochondrial protein was added to each chamber. 20 μL of 1 M sodium pyruvate was added, resulting in a final concentration of 10 mM. 4 μL of 1 M malate was added for a final concentration of 2 mM. 8 μL of 0.5 M ADP was added for a final concentration of 2 mM. 2 μL of 1mM oligomycin was added for a final concentration of 1 μM. 10 μL of 100 μM FCCP was added for a final concentration of 0.5 μM. 2 μL of 1 mM rotenone were added for a final concentration of 1 μM. 2 μL of 1 mM antimycin A were added for a final concentration of 1 μM.

2.2 Translation into a Mathematical Model

2.2.1 Modelling Strategy

The experimental data available are oxygen concentration and oxygen consumption rate, which represent the activity of the mitochondria. Since oxygen consumption occurs at complex IV, the focus here is on modelling this complex. However, complex IV cannot function in isolation. Complexes I, III, and V control proton movement across the inner mitochondrial membrane, which directly affects the activity of complex IV. Complex IV also requires delivery of the reduced form of cytochrome c, which is produced by complexes I - III. Thus the end product of complexes I-III functions as input into complex IV, which allows for a description of the set of complexes I-III as a single input. Modelling the function of complexes I-III, complex IV and complex V thereby constitutes the *baseline model* shown below in system (2.9). Furthermore, incorporating the effects of the reagents described above, *oligomycin*, *FCCP*, *Antimycin A/rotenone*, produces three additional systems shown in (2.10), (2.14), and (2.15), respectively.

These developed models are kinetic descriptions of the chemical reactions in the mitochondria. The reaction driven by complex IV is described previously in (1.7), but since complexes I-III are described as a unique subsystem in the model, the previously described chemical reactions (1.2) - (1.6) sum to:

$$complex\ I-III: \quad 2Cytc_{ox} + NADH_2 + 6H_N^+ \longrightarrow NAD^+ + 2Cytc_{red} + 8H_P^+ \tag{2.1}$$

Thus, the four substrates pertinent to the models are:

- cytochrome c reduced, where its concentration at time t is represented by the state variable $r(t)$,
- oxygen, where its concentration at time t is represented by the state variable $o(t)$,

- matrix protons H_N^+, where its concentration at time t is represented by the state variable $\omega(t)$,
- inter-membrane space protons H_P^+, where its concentration at time t is represented by the state variable $\rho(t)$.

Since it is assumed that $NADH^+$ is abundant in the system, NAD and $NADH^+$ are not explicitly described in the model.

The general framework for the rate of change of concentration for the four main substrates takes the form:

$$\begin{aligned} \frac{dr}{dt} &= 2F_0 - 2F_4 \\ \frac{do}{dt} &= -\frac{1}{2}F_4 \\ \frac{d\omega}{dt} &= -6F_0 - 4F_4 + F_5 \\ \frac{d\rho}{dt} &= 8F_0 + 2F_4 - F_5 \end{aligned} \tag{2.2}$$

The functional F_i represents the reaction speed of each reaction: F_0 describes complex I-III activity, F_4 describes complex IV activity, and F_5 describes complex V activity. The fundamental assumption for these reaction speeds is that protein complexes are modelled as enzymes. The specifics of each reaction speed are detailed in the following sections.

2.2.2 F_0: Complexes I - III

The rate equation for the compound input F_0 can be determined by using simple enzyme kinetic derivation on a single substrate. This reaction can then be modelled as a simple enzymatic reaction catalyzing the reduction of cytochrome c oxidized. Assuming simple Michaelis-Menten kinetics as derived in Appendix A.1.2, the reaction speed generally follows:

$$v = \frac{V_{max}S}{K_m + S} \tag{2.3}$$

where V_{max} is the maximum speed of the reaction, S is the substrate, and K_m is the concentration of substrate at which the reaction achieves half the maximum speed, known

as the *Michaelis-Menten constant*. It is assumed that substrate S for the the combined complexes I to III is cytochrome c oxidized. Furthermore, since cytochrome c is conserved in the system, the concentration of cytochrome c oxidized is obtained from the total concentration of cytochrome c c_0 and cytochrome c reduced r, such that $S = cyt\, c_{ox} = c_0 - r$.

Additionally, since the ratio of protons on either side of the membrane, ω and ρ, will influence the rate of electron transport chain activity, the balance of protons must scale the function of equation (2.3). Under ideal circumstances, the membrane potential of the inner mitochondrial membrane $\Delta\Psi$, but this must be determined experimentally, which is not available in the current study. Thus, to approximate the effect of membrane potential $\Delta\Psi$, the pH ratio $\frac{\omega}{\rho}$ is used. Since protons existing in the mitochondrial matrix, ω, would drive the ETC forward if $\omega > \rho$, then v can be scaled by multiplying by $\frac{\omega}{\rho}$ to represent the effect this balance will have on ETC function. This ratio approximates the effect of the *pmf* established by the migration of protons from the N side to the P side, thus affecting the activity of the ETC.

Altogether, this gives a rate equation for complexes I-III as:

$$F_0(r, \omega, \rho) = \left(\frac{V_{max_{cI-III}}(c_0 - r)}{K_{m_{cI-III}} + (c_0 - r)}\right)\left(\frac{\omega}{\rho}\right). \tag{2.4}$$

2.2.3 F_4: Complex IV

Deriving the rate equation for complex IV assumes that it follows activation enzyme kinetics, detailed in Appendix A.1.4. This is a valid assumption for complex IV since cytochrome c reduced must first be bound to complex IV before oxygen can bind [27], thus illustrating the *activating* effect cytochrome c reduced has on complex IV's processing of oxygen. For activation enzyme kinetics the general rate equation is:

$$v = \frac{V_{max} S^b}{K_m\left(1 + \frac{K_1}{A^a}\right) + S^b} \tag{2.5}$$

where S represents the enzyme substrate with stoichiometric constant b, and A represents the enzyme activator with stoichiometric constant a. For complex IV, the substrate is oxygen and thus $S = o$, while the activator is cytochrome c reduced, so $A = r$, with stoichiometric constants b and a initially determined by reaction (1.7). However, since the reactions occur in a parallel manner and result in simultaneous binding of all required substrates [1], both a and b can approximate binding availability by assuming that $a = b = 1$.

Finally, as with complexes I-III, since the activity of the ETC is modulated by proton balance across the membrane, equation (2.5) is also modulated by the ratio $\frac{\omega}{\rho}$. Thus, the rate equation for complex IV is:

$$F_4(r, o, \omega, \rho) = \left(\frac{V_{max_{cIV}} o}{K_{m_{cIV}} \left(1 + \frac{K_{cIV}}{r}\right) + o} \right) \left(\frac{\omega}{\rho} \right) \tag{2.6}$$

2.2.4 F_5: Complex V

An existing model of complex V activity was used to describe the rate of complex V in the current model. The model developed by Jain and Nath [28] was derived by a similar derivation as in Michaelis-Menten kinetics, but with additional considerations for pH and the ratio of protons across the inner mitochondrial membrane. Jain and Nath modelled complex V with the following equation for reaction speed:

$$v = \left[\frac{(k_s k_r E_0) H_P^+}{H_P^+ + \left(\frac{K_1}{K_2}\right) H_N^+ + \left(K_1 + \frac{k_r E_0}{k_t}\right)} \right] \tag{2.7}$$

Since the biological conditions in their experiment and the experiments used for the current model are different, constant terms such as k_s, k_r and E_0 were instead modelled as parameters in the current model. Additionally, the units for v in equation (2.7) is s^{-1}. Since the reaction speeds in the current model are nmol/(mL s), and complex V is directly modulated by the level of ρ protons, this rate equation is also scaled by ρ concentration in the current model. This corrects the units of v to nmol/(mL s).

With these slight modifications to equation (2.7), the equation for F_5 used in the current model is:

$$F_5(\omega, \rho) = \left(\frac{V_{max_{cV}} \rho}{\rho + K_{cV}\omega + K_{m_{cV}}} \right) \rho \tag{2.8}$$

where $V_{max_{cV}} = k_s k_r E_0$, $K_{cV} = \frac{K_1}{K_2}$, and $K_{m_{cV}} = K_1 + \frac{k_r E_0}{k_t}$.

2.2.5 The Baseline Model

Using the forms of F_i developed in the previous sections, and the general framework for the model given in subsystem (2.2), the equations for the baseline system are collected together below in subsystem (2.9). Thus, substituting Equations (2.4), (2.6) and (2.8) into subsystem (2.2) yields:

$$\text{Baseline} \begin{cases} \frac{dr}{dt} = 2\left(\frac{V_{max_{cI-III}}(c_0 - r)}{K_{m_{cI-III}} + (c_0 - r)}\right)\left(\frac{\omega}{\rho}\right) - 2\left(\frac{V_{max_{cIV}} o}{K_{m_{cIV}}\left(1 + \frac{K_{cIV}}{r}\right) + o}\right)\left(\frac{\omega}{\rho}\right) \\ \frac{do}{dt} = -\frac{1}{2}\left(\frac{V_{max_{cIV}} o}{K_{m_{cIV}}\left(1 + \frac{K_{cIV}}{r}\right) + o}\right)\left(\frac{\omega}{\rho}\right) \\ \frac{d\omega}{dt} = -6\left(\frac{V_{max_{cI-III}}(c_0 - r)}{K_{m_{cI-III}} + (c_0 - r)}\right)\left(\frac{\omega}{\rho}\right) - 4\left(\frac{V_{max_{cIV}} o}{K_{m_{cIV}}\left(1 + \frac{K_{cIV}}{r}\right) + o}\right)\left(\frac{\omega}{\rho}\right) \\ \qquad + \left(\frac{V_{max_{cV}} \rho}{\rho + K_{cV}\omega + K_{m_{cV}}}\right)\rho \\ \frac{d\rho}{dt} = 8\left(\frac{V_{max_{cI-III}}(c_0 - r)}{K_{m_{cI-III}} + (c_0 - r)}\right)\left(\frac{\omega}{\rho}\right) + 2\left(\frac{V_{max_{cIV}} o}{K_{m_{cIV}}\left(1 + \frac{K_{cIV}}{r}\right) + o}\right)\left(\frac{\omega}{\rho}\right) \\ \qquad - \left(\frac{V_{max_{cV}} \rho}{\rho + K_{cV}\omega + K_{m_{cV}}}\right)\rho \end{cases} \tag{2.9}$$

This system of equations represents the baseline subsystem, prior to injection of any chemical reagents outlined in the experimental protocol.

2.2.6 Modelling Changes to Oxidative Phosphorylation

The experimental protocol involves injection of four chemical reagents as discussed in Sections 1.1.3 and 2.1.2. Each of these reagents affect the operation of the mitochondria, and thus require adaptation of the baseline subsystem shown above in Equation (2.9).

Oligomycin, the first injection reagent, disrupts the activity of the proton channel in complex V by binding to the ring of c subunits in this enzyme. This results in the flux of protons dropping to zero, implying that $F_5 \to 0$. Therefore, in this condition F_5 is 0, yielding:

$$\text{Oligomycin} \begin{cases} \frac{dr}{dt} &= 2\left(\frac{V_{max_{cI-III}}(c_0-r)}{K_{m_{cI-III}}+(c_0-r)}\right)\left(\frac{\omega}{\rho}\right) - 2\left(\frac{V_{max_{cIV}}o}{K_{m_{cIV}}\left(1+\frac{K_{cIV}}{r}\right)+o}\right)\left(\frac{\omega}{\rho}\right) \\ \frac{do}{dt} &= -\frac{1}{2}\left(\frac{V_{max_{cIV}}o}{K_{m_{cIV}}\left(1+\frac{K_{cIV}}{r}\right)+o}\right)\left(\frac{\omega}{\rho}\right) \\ \frac{d\omega}{dt} &= -6\left(\frac{V_{max_{cI-III}}(c_0-r)}{K_{m_{cI-III}}+(c_0-r)}\right)\left(\frac{\omega}{\rho}\right) - 4\left(\frac{V_{max_{cIV}}o}{K_{m_{cIV}}\left(1+\frac{K_{cIV}}{r}\right)+o}\right)\left(\frac{\omega}{\rho}\right) \\ \frac{d\rho}{dt} &= 8\left(\frac{V_{max_{cI-III}}(c_0-r)}{K_{m_{cI-III}}+(c_0-r)}\right)\left(\frac{\omega}{\rho}\right) + 2\left(\frac{V_{max_{cIV}}o}{K_{m_{cIV}}\left(1+\frac{K_{cIV}}{r}\right)+o}\right)\left(\frac{\omega}{\rho}\right) \end{cases} \tag{2.10}$$

The next reagent injected experimentally is $FCCP$, which re-enables OCR by uncoupling complex V from the ETC. This occurs by the ionophore activity of $FCCP$, which creates a artificial proton channels in the membrane, as discussed in Section 1.1.3.

The flow of protons due to injection of FCCP must be modelled using the Nernst-Planck equation [29], which models the flow of a charged chemical species through a continuous fluid. This equation is given by:

$$J = -D(\nabla c + \frac{zF}{RT}c\Delta\Psi) \tag{2.11}$$

where J is the *flux* of flow of the compound over a certain area, D is its diffusion coefficient, c its concentration, z is its charge, F is Faraday's constant, R is the universal gas constant, T is the absolute temperature of the system, and Ψ is the membrane potential. Assuming that the protons are moving toward equilibrium or the *nernst potential* of H^+ [29], which is given by:

$$\Delta\phi = \frac{RT}{zF}ln\left(\frac{\rho}{\omega}\right), \tag{2.12}$$

this potential can then be applied to $\Delta\phi$ in equation (2.11) yielding:

$$F_6 = \frac{D_{H^+}}{A}\Big((\rho-\omega) + \rho\ln\left(\frac{\rho}{\omega}\right)\Big) \tag{2.13}$$

where D_{H^+} is the diffusion coefficient for a proton, and A is the area over which we are measuring the flux of protons. Since this area is variable and unknown, this model will estimate the coefficient $\frac{D_{H+}}{A}$ as a single model parameter, namely parameter D_h. Equation (2.13) represents the rate of proton movement toward the mitochondrial matrix.

It is noteworthy to mention that when F_5 is inhibited by *oligomycin*, $F_6 \to 0$ at equilibrium since at this point proton concentrations balance across the membrane and there will be no net flux of protons.

Thus, following FCCP injection, the subsystem becomes:

$$\text{FCCP}\begin{cases} \frac{dr}{dt} &= 2\left(\frac{V_{max_{cI-III}}(c_0-r)}{K_{m_{cI-III}}+(c_0-r)}\right)\left(\frac{\omega}{\rho}\right) - 2\left(\frac{V_{max_{cIV}}o}{K_{m_{cIV}}\left(1+\frac{K_{cIV}}{r}\right)+o}\right)\left(\frac{\omega}{\rho}\right) \\ \frac{do}{dt} &= -\frac{1}{2}\left(\frac{V_{max_{cIV}}o}{K_{m_{cIV}}\left(1+\frac{K_{cIV}}{r}\right)+o}\right)\left(\frac{\omega}{\rho}\right) \\ \frac{d\omega}{dt} &= -6\left(\frac{V_{max_{cI-III}}(c_0-r)}{K_{m_{cI-III}}+(c_0-r)}\right)\left(\frac{\omega}{\rho}\right) - 4\left(\frac{V_{max_{cIV}}o}{K_{m_{cIV}}\left(1+\frac{K_{cIV}}{r}\right)+o}\right)\left(\frac{\omega}{\rho}\right) \\ &\quad +D_h\Big((\rho-\omega)+\rho\, ln\Big(\frac{\rho}{\omega}\Big)\Big) \\ \frac{d\rho}{dt} &= 8\left(\frac{V_{max_{cI-III}}(c_0-r)}{K_{m_{cI-III}}+(c_0-r)}\right)\left(\frac{\omega}{\rho}\right) + 2\left(\frac{V_{max_{cIV}}o}{K_{m_{cIV}}\left(1+\frac{K_{cIV}}{r}\right)+o}\right)\left(\frac{\omega}{\rho}\right) \\ &\quad -D_h\Big((\rho-\omega)+\rho\, ln\Big(\frac{\rho}{\omega}\Big)\Big) \end{cases} \tag{2.14}$$

When *rotenone* and *antimycin a* are injected, both complex I and complex III are competitively inhibited. Complex I is inhibited by *rotenone* while complex III is inhibited by *antimycin a*. Inhibition of complexes I and III result in $F_0 \equiv 0$, yielding the system:

$$\text{Inhibition}\begin{cases} \frac{dr}{dt} &= -2\left(\frac{V_{max_{cIV}}o}{K_{m_{cIV}}\left(1+\frac{K_{cIV}}{r}\right)+o}\right)\left(\frac{\omega}{\rho}\right) \\ \frac{do}{dt} &= -\frac{1}{2}\left(\frac{V_{max_{cIV}}o}{K_{m_{cIV}}\left(1+\frac{K_{cIV}}{r}\right)+o}\right)\left(\frac{\omega}{\rho}\right) \\ \frac{d\omega}{dt} &= -4\left(\frac{V_{max_{cIV}}o}{K_{m_{cIV}}\left(1+\frac{K_{cIV}}{r}\right)+o}\right)\left(\frac{\omega}{\rho}\right) + D_h\Big((\rho-\omega)+\rho\, ln\Big(\frac{\rho}{\omega}\Big)\Big) \\ \frac{d\rho}{dt} &= 2\left(\frac{V_{max_{cIV}}o}{K_{m_{cIV}}\left(1+\frac{K_{cIV}}{r}\right)+o}\right)\left(\frac{\omega}{\rho}\right) - D_h\Big((\rho-\omega)+\rho\, ln\Big(\frac{\rho}{\omega}\Big)\Big) \end{cases} \tag{2.15}$$

2.2.7 Parameter Estimation

The model contains eight parameters within the provided set of three equations shown in system (2.9) and an additional parameter due to the introduction of FCCP as shown in equation (2.13). Finally, initial concentrations for cytochrome c reduced and oxidized were also optimized as parameters. Thus, the full set of parameters is represented uniquely by the 11-vector:

$$\vec{p} = \left[V_{max_{cI-III}}, K_{m_{cI-III}}, V_{max_{cI-III}}, K_{m_{cIV}}, K_{cIV}, V_{max_{cV}}, K_{m_{cV}}, K_{cV}, D_h, Cyt\ c_{red}(0), Cyt\ c_{ox}(0)\right] \quad (2.16)$$

To estimate the set of parameter values $\vec{p}$ that provide the closest approximation to the experimental data, a process referred to as *calibration*, each parameter in $\vec{p}$ was independently varied and used by the model to simulate oxygen concentration over time. The error between model output and the experimental oxygen concentration data was then minimized. This was done by finding the $\vec{p}$ that produced the smallest *mean-squared error* (MSE) calculated as:

$$E(\vec{p}) = \frac{1}{n}\sum_{i=1}^{n}\left(o(t_i, \vec{p}) - x(t_i)\right)^2 \quad (2.17)$$

where $o(t_i, \vec{p})$ is the concentration of oxygen evaluated by the model at time t_i using the parameter set $\vec{p}$, and $x(t_i)$ is the experimental concentration of oxygen at the time t_i.

Parameter values in each evaluated $\vec{p}$ were systematically varied using a *genetic algorithm* run on the MATLAB R2014a platform. A genetic algorithm is an optimization technique that mimics the principles of genetic mutations and DNA crossovers to introduce parameter value variations in a set of parameters $\vec{p}$. The general implementation of the genetic algorithm outlined in Appendix D was applied as follows:

1. Create an initial population of n solvers, where each solver represents a unique $\vec{p}$. In this project, two parallel populations were created each with size $n = 125$.

2. Give each solver an initial set of m parameter values in the form $\{p1, p2, p3, ..., p_m\}$. Thus each solver started with the same $\vec{p}$ values.

3. Randomly change the values of some of the parameters of each solver to create variation in the initial population. Changes were bounded according to Table 2.1, where bounds were approximated by using literature values of similar parameter values as listed in Table 3.1.

4. Evaluate the MSE for each $\vec{p}$.

5. Allow the 10% of solvers that had the smallest MSE's to proceed to the next generation.

6. Introduce partial interchanges of matching parameters between two random solvers in the top 10%. This entailed decreasing the value(s) from one randomly selected top solver, and increasing the value(s) of the same parameter(s) in another randomly selected top solver by the same amount. This interchange occurred between top solvers from the same population at a rate of 0.1, or between one top solver from each population at a probability of 0.25. Interchanges continued until two new populations were created with size 125, including the top 10% of solvers from the previous generation.

7. Randomly introduce alterations to parameter values within the new populations. The probability of this occurring was 0.1.

8. Repeat the process from step 4 until the smallest MSE remains the same, for the same $\vec{p}$ for 100 generations. The $\vec{p}$ that produced this MSE then became the best parameter set, designated $\vec{p^*}$, and the genetic algorithm exits.

The software used to carry out the genetic algorithm outlined here was developed by, and used with permission from, Dr. Jason Fiege from the Department of Physics and Astronomy, University of Manitoba.

This process was used to find a global minimum MSE, or smallest error. The parameter set resulting in this smallest error was designated $\vec{p^*}$.

2.2.7.1 Calibrating Multiple Conditions

The subsystems detailed in Section 2.2.6 refer to separate systems of equations depending on the reagents in the cellular environment. In the experimental protocol, injection points are

Table 2.1: Bounds for each parameter during parameter estimation. Bounds for the values of each parameter in the parameter set $\vec{p}$. Each parameter was allowed to vary within these bounds, inclusively, during the operation of the genetic algorithm. Bounds were largely determined based on literature values for each parameter, as listed in Table 3.1, and then expanded to allow for a more thorough investigation through the parameter space.

Parameter	Lower Bound	Upper Bound	Units
$V_{max_{cI\text{-}III}}$	1×10^{-2}	1×10^{4}	nmol/mL s
$K_{m_{cI\text{-}III}}$	1×10^{-3}	1×10^{5}	nmol/mL
$V_{max_{cIV}}$	1×10^{-2}	1×10^{4}	nmol/mL s
$K_{m_{cIV}}$	1×10^{-3}	1×10^{5}	nmol/mL
K_{cIV}	1×10^{-2}	1×10^{4}	$nmol^{-1}s^{-1}$
$V_{max_{cV}}$	1×10^{-2}	1×10^{4}	nmol/mL s
$K_{m_{cV}}$	1×10^{-3}	1×10^{5}	nmol/mL
K_{cV}	1×10^{-2}	1×10^{4}	-
D_h	1×10^{-2}	1×10^{5}	cm^2/s
Cyt c_{red}	1×10^{-2}	1×10^{3}	nmol/mL
Cyt c_{ox}	1×10^{-2}	1×10^{3}	nmol/mL

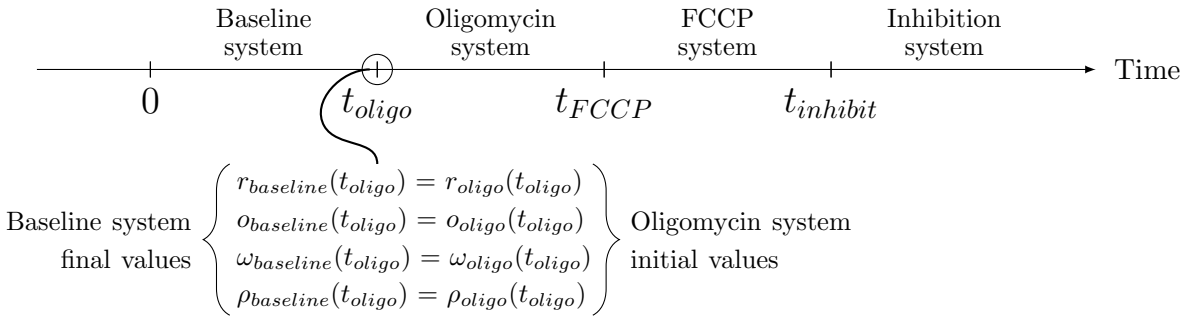
introduced without interruption of the system. Thus, each condition is continuous with the next. In calibrating the system, the Runge-Kutta integration technique was used to integrate each system of equations by using the previous system's final state values as initial conditions for the next. For instance, Fig. 2.1 shows that the final values for $r(t)$, $o(t)$, $\omega(t)$, and $\rho(t)$ in the baseline subsystem are provided as initial values into the subsequent *oligomycin* subsystem. This was applied to each transition time point t_{oligo}, t_{FCCP}, and $t_{inhibit}$, allowing for full simulation of the experimental protocol without discontinuity.

This approach was used to simultaneously calibrate the four systems developed to determine parameters globally consistent with the experimental protocol. Time points of integration were matched to experimental data across all conditions, and equation (2.17) was minimized for time points spanning all conditions simultaneously.

2.2.7.2 Sensitivity Analysis

Following the method outlined in [14], the sensitivity of each parameter given finite changes in parameter values were determined by calculating the maximum error due to a 10% change in a

Figure 2.1: Representation of the calibration strategy used. The final values of each subsystem are used as initial conditions for evaluation of the next system. Illustrated here is the usage of the final concentrations of r, o, ω, and ρ from evaluation of the baseline system as initial conditions in the oligomycin system. This strategy is used to transition between subsystems at each injection time point.



specific parameter value. Thus, sensitivity coefficients were determined for each parameter as follows:

$$S_i = max\left(\frac{|E(p_i^* \pm 0.1p_i^*) - E(\vec{p^*})|}{0.1E(\vec{p^*})}\right) \tag{2.18}$$

where $E(\vec{p^*})$ is the minimum MSE as calculated by equation (2.17), p_i^* is the optimal value of the ith parameter in $\vec{p^*}$, and $i = \{1, 2, 3, ..., 9\}$. The term $E(p_i^* \pm 0.1p_i^*)$ is an evaluation of equation (2.17) after deviating p_i^* by 10 % both above and below the estimated value.

Chapter 3

Results

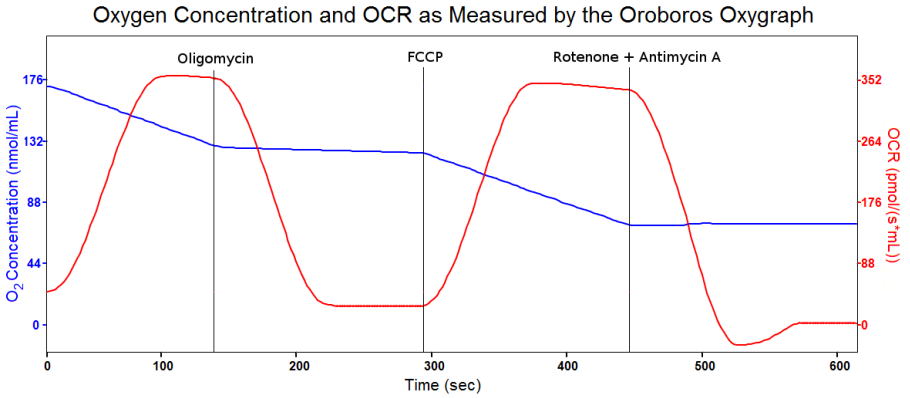
3.1 Biological Data

Experimental data was acquired using both the Seahorse XF24 Analyzer and the Oroboros Oxygraph-2k.

Fig. 3.1 depicts a representative graph of oxygen concentration versus time and OCR versus time as measured by the Oroboros Oxygraph-2k. The blue line represents oxygen concentration, whereas the red line represents OCR. Both experimental protocols described in section 2.1 represent data collected for baseline, *oligomycin*, $FCCP$, and *rotenone*/*Antimycin A* conditions.

Figures 3.3 and 3.2 depict representative graphs of oxygen consumption rate (OCR) versus time and oxygen concentration versus time as measured by the Seahorse XF24 Analyzer. Evident in the graph of OCR measured by the XF24 in Fig. 3.3, maximal OCR during the $FCCP$ condition is much more pronounced than that measured in the oxygraph condition in Fig. 3.1. Since the measurements in the XF24 were recorded using cells as described in Section 2.1.2, some substances required for optimal baseline activity are not supplemented as they are in the Oxygraph protocol as detailed in Section 2.1.3. This is largely a result of the permeability of the plasma membrane acting as a barrier to certain substances. Given this difference, and the operation of the XF24's plunger during recording (discussed below in Section 4.1), the data acquired from the XF24 was not used to calibrate the model quantitatively. Instead, the response

Figure 3.1: Experimental oxygen concentration and OCR data from the Oroboros Oxygraph-2k. The blue line shows oxygen concentration data, whereas the red line shows OCR data. Measurements were taken in cortical mitochondria from CD1 mice following the experimental protocol outlined in Section 2.1.2.



of the mitochondria to the injected reagents, *oligomycin*, *FCCP*, *rotenone*, and *antimycin A* was validated by ensuring similar responses are acquired between the XF24 and Oxygraph measurements. Given the red curve shown in Fig. 3.1, representing OCR measurements, the response of mitochondria is similar to that in the XF24 in Fig. 3.3.

Figure 3.2: Experimental oxygen concentration data from the Seahorse XF24 Analyzer. Measurements were carried out using cultured cortical neurons from CD1 mice. Oxygen concentration is measured in four different conditions: under baseline functioning, following *oligomycin* injection, following *FCCP* injection, and following injection of *rotenone* and *antimycinA*. Although the data appears oscillatory, this is a result of the measurement plunger lifting between measurement intervals, allowing each measurement space to mix with the entire well. The details and consequences of this measurement procedure are discussed in Section 4.1.

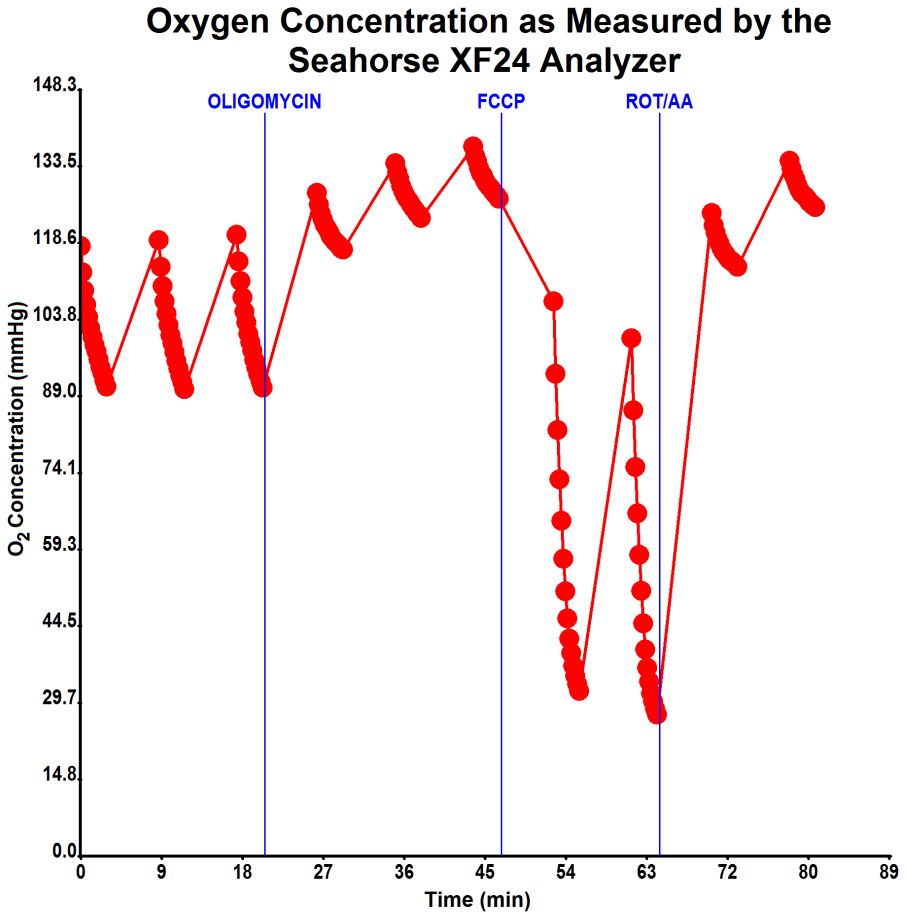
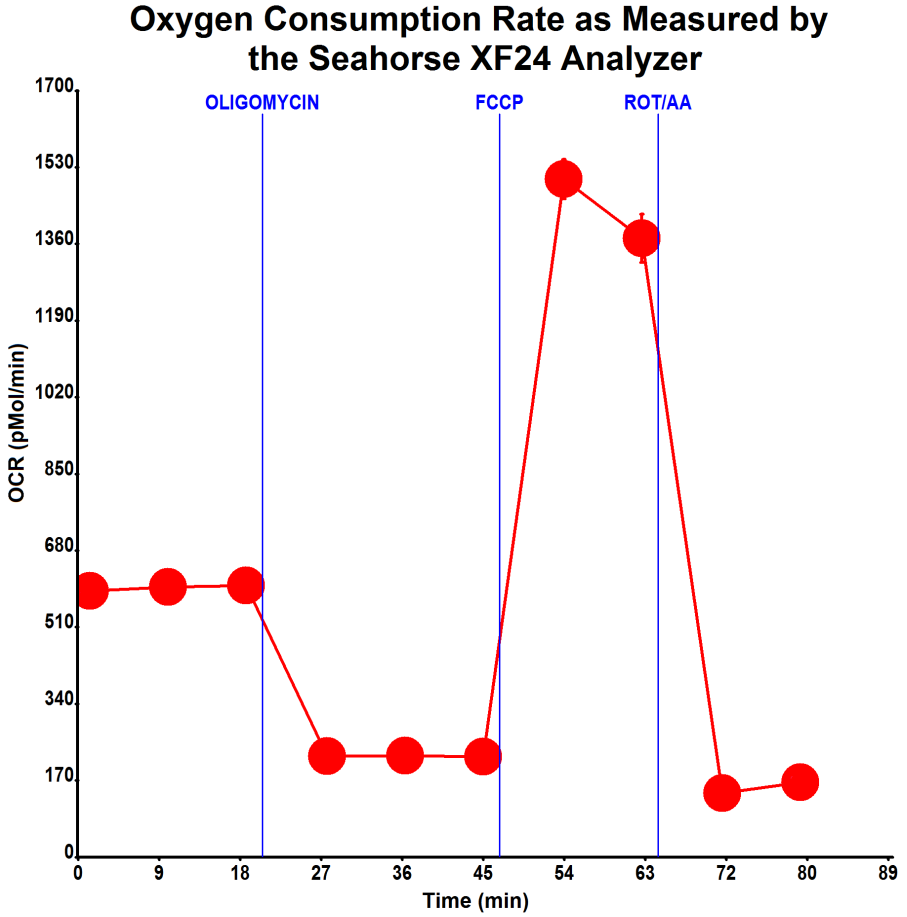


Figure 3.3: Experimental oxygen consumption rate data from the Seahorse XF24 Analyzer. Measurements were carried out using cultured cortical neurons from CD1 mice. OCR is measured in four different conditions: under baseline functioning, following *oligomycin* injection, following $FCCP$ injection, and following injection of *rotenone* and *antimycinA*. In cultured cells, the response of the system to $FCCP$ is pronounced compared to that of baseline, although this is due limited availability of certain biological substrates in cells but not found in tissue due to the permeability of the plasma membrane as a barrier. This is further discussed in Section 4.1.

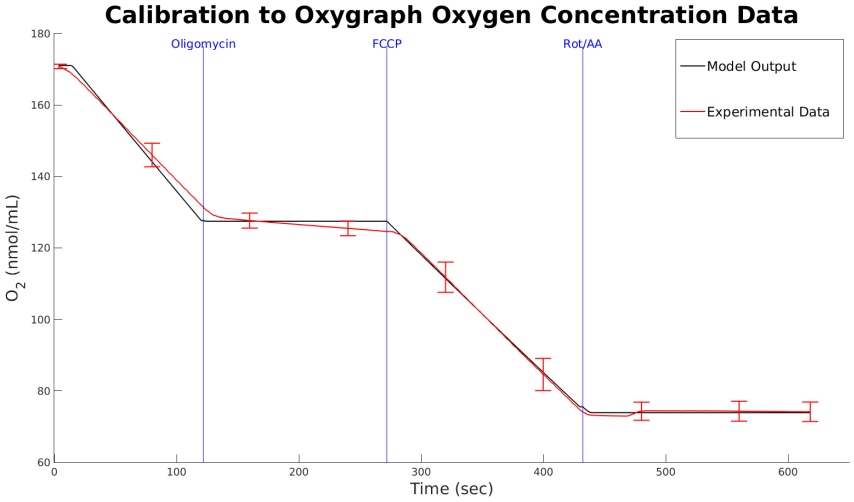


3.2 Model

3.2.1 Parameter Estimation

The model was calibrated to oxygen concentration data by minimizing equation (2.17), yielding a minimum value of $E(\vec{p^*}) = 1.664$ after $n = 1311$ generations. Fig 3.4 shows a plot of the experimental oxygen concentration data alongside the output of oxygen concentration $o(t)$.

Figure 3.4: Calibration of the model to experimental oxygen concentration data. The red line shows the experimental oxygen concentration data as measured in the Oxygraph-2k, whereas the black line shows the simulated oxygen concentration over time as output by the model. The MSE of the fit between simulated and measured oxygen concentration was $E(\vec{p^*}) = 1.664$, calibrated by using the genetic algorithm outlined in Section 2.2.7.



It is evident from this figure that the model accurately predicts oxygen concentration levels as measured by the Oxygraph-2k. In the baseline condition, the model produced stable, constant consumption of oxygen, consistent with the Oxygraph data and with a constant OCR shown in the Seahorse data.

In the oligomycin condition, although the model predicts no consumption of oxygen, there is some transient consumption of oxygen shown in the Oxygraph data. This may be interpreted as a result of proton leak allowing protons to move through the membrane without flowing through complex V. These protons then react with oxygen resulting in very slow, but non-zero consumption of oxygen. The model does not incorporate a representation for proton leak, which

may explain the slight disparity in the oligomycin condition. However, the model provides an accurate representation of the Oxygraph data, and also qualitatively matches the effect obtained in the Seahorse data.

In the FCCP condition, the model predicts a constant consumption of oxygen, which fits the pattern of oxygen consumption in the Oxygraph data and a constant OCR shown in the Seahorse data. Finally, the rotenone/antimycin A condition dictates that there will be no oxygen consumption due to complete inhibition of the system, which the model predicts. Thus each condition is accurately represented by the model, both quantitatively as per calibration to the Oxygraph data, and qualitatively as per comparison to the Seahorse data.

The output of the model for the system with initial conditions as defined by the experimental protocol is shown below in Fig. 3.5.

Figure 3.5: Output for the calibrated model. These graphs show simulated oxygen concentration, OCR, cytochrome c reduced, matrix proton concentration, and intermembrane space (IMS) proton concentration over time. The calibrated parameter values for this simulation are listed in Table 3.1.

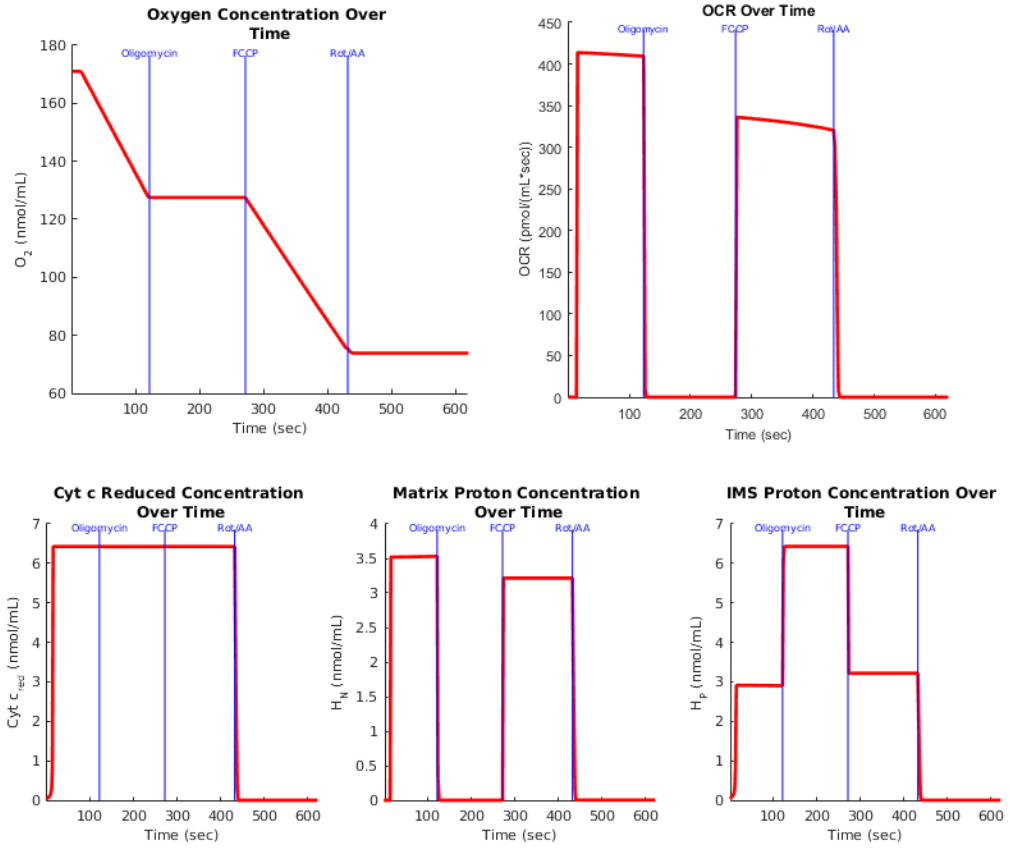


Table 3.1: Calibrated model parameter values. Estimated values for the best parameter set $\vec{p}^*$, as determined using the genetic algorithm and calibration approach described in Section 2.2.7. Literature values listed here were also used to determine approximate ranges of calibration for each parameter as listed in Table 2.1.

Parameter	Description	Estimated Value	Literature Value	Units
$V_{max_{cI\text{-}III}}$	Maximum reaction velocity of F_0	9812.4	[a] -	nmol/mL s
$K_{m_{cI\text{-}III}}$	Michaelis-Menten reaction constant for F_0	12.251	[a] -	nmol/mL
$V_{max_{cIV}}$	Maximum reaction velocity of F_4	0.7225	37.33 - 40.47 [19]	nmol/mL s
$K_{m_{cIV}}$	Michaelis-Menten reaction constant for F_4	0.1627	7 - 10 [19][30]	nmol/mL
K_{cIV}	Equilibrium constant for F_4	365.02	[a] -	$nmol^{-1}s^{-1}$
$V_{max_{cV}}$	Maximum reaction velocity of F_5	43166	57 [28]	nmol/mL s
$K_{m_{cV}}$	Michaelis-Menten reaction constant for F_5	11388	6607 [28]	nmol/mL
K_{cV}	Ratio of $\frac{K_1}{K_2}$ in complex V	9342.6	900 [28]	-
D_h	Diffusion coefficient of hydrogen per unit area	5116.1	[b] 5.8×10^{-5}/A [31]	s^{-1}
cyt $c_{red}(0)$	Initial concentration of cytochrome c in reduced form	0.0500	-	nmol/mL
cyt $c_{ox}(0)$	Initial concentration of cytochrome c in oxidized form	6.3766	-	nmol/mL
c_0	Total initial cytochrome c	6.4266	[c] -	nmol/mL

[a]These values are novel to the current model, thus no reference values exist.
[b]This reference value regards diffusion D_{H^+} from [31], whereas D_h regards $\frac{D_{H+}}{A}$, diffusion normalized to an area A, which is not measureable in the current experimental protocol.
[c]This value is set by the initial values of the model.

Table 3.1 shows the estimated parameter values upon calibration of the model to experimental data. For each parameter common values ranges for the given parameter and their units are listed. Literature values listed in Table 3.1 were acquired experimentally in brain tissue, but with differing biological conditions a range of common values for complex IV kinetics is provided. Ranges for common values are dependent on the properties of the mitochondria, such as availability of oxygen [32], source of tissue [19], cellular location of mitochondria [33], or age [33][34]. The literature values listed were used to construct the value ranges for each parameter during calibration, as outlined in Section 2.2.7.

The parameter values estimated for the current model are slightly lower than found in the literature, particularly for the maximal velocity of complex IV. The model parameters for the adopted function F_5, however, show much higher values than in the original model [28]. This is likely due to the constraints placed on pH values, discussed below in Section 3.2.2, which were not carried out in the original complex V model [28]. Jain and Nath rather illustrated the effect of varying pHs and the pH gradient between the matrix and IMS, which resulted in the current model's pH values (IMS pH = 7.4 and matrix pH = 7.8) and pH gradient (ΔpH

$= -0.4$) showing low levels of complex V activity. Thus, the increased parameter values in the current model may reflect a compensatory response to adequately describe an appropriate level of complex V functioning given the assumed pH values.

As more experimental data is acquired, specifically by measuring the dynamics of the system aside from just oxygen concentration, the ideal parameter set $\vec{p^*}$ may represent values expressed in the literature more closely.

3.2.2 Robustness of the Model

Initial conditions are inherently set in the experimental protocol of both the Seahorse XF24 Analyzer and the Oroboros Oxygraph. Altering these values, however, can provide an estimate for the behavior and limitations of the model in comparison to biological expectations. Alterations to initial conditions are run in the baseline model over an extended time frame ($t \to 2000$ s) to show the output of the model without introduction of chemical reagents.

The altered initial conditions used for testing robustness of the model are summarized in Table 3.2. High and low concentration values were determined by multiplying baseline concentration values by a factor of 10^3 or 10^{-3}, respectively. Each figure shown in Figs. 3.6 - 3.13 show two simulations of the model, where the black lines represent output of the model under normal initial concentrations, and red lines represent output of the model under altered initial conditions.

Table 3.2: Summary of the output during the tests of robustness. Each primary substrate in the model was either increased by a factor of 10^3 or decreased by a factor of 10^{-3}. The resulting model outputs are shown in Figs. 3.6–3.13, and are listed here with the accompanying change to initial conditions.

Substrate Changed	**High Concentration (nmol/mL)**	**Low Concentration (nmol/mL)**	**Model Concentration (nmol/mL)**
Initial cytochrome c reduced concentration	50 (Fig. 3.6)	5.0×10^{-5} (Fig. 3.7)	0.0500
Initial oxygen concentration	1.71055×10^5 (Fig. 3.8)	0.171055 (Fig. 3.9)	171.06
Initial matrix proton concentration	15.85 (Fig. 3.10)	1.585×10^{-5} (Fig. 3.11)	0.01585
Initial Intermembrane space (IMS) proton concentration	39.81 (Fig. 3.12)	3.981×10^{-5} (Fig. 3.13)	0.03981

The output of the model under normal conditions have initial concentrations of $r(0) = 0.0500$ nmol/mL, $o(0) = 171.05$ nmol/mL, $\omega(0) = 0.01585$ nmol/mL, and $\rho(0) = 0.03981$ nmol/mL. The initial concentration of cytochrome c reduced was acquired through optimization, yielding $r(0) = 0.0500$ nmol/mL. Experimental data dictated an initial concentration of oxygen to be $o(0) = 171.05$ nmol/mL. Since a pH of 7.4 was determined experimentally, this pH dictated $\rho(0) = 0.03891$, as described in [1]. Finally, assuming a resting pH gradient of -0.4 pH units [35][36][37], with the matrix compartment less acidic than the IMS, this yielded an initial concentration of matrix protons to be $\omega(0) = 0.0158$ nmol/mL, or a pH of 7.8 in the matrix.

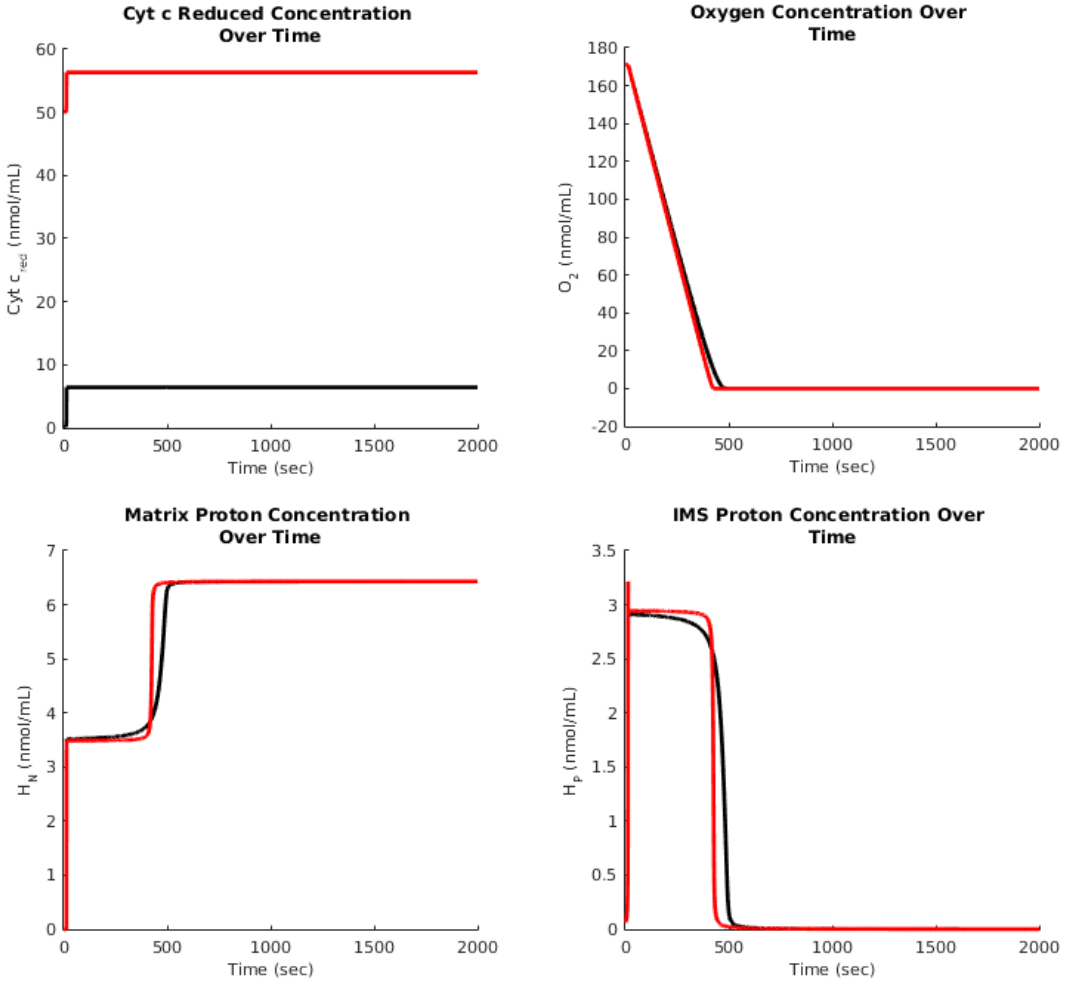
It is expected that under normal conditions, oxygen will be fully consumed, thus $o(t) \to 0$ as $t \to \infty$. This will result in stabilization of proton flow, but since complex V is still operational, matrix protons will equilibrate to a final proton concentration ω_f and IMS protons will translocate into the matrix, resulting in $\rho(t) \to 0$ as $t \to \infty$. Since cytochrome c reduced will no longer be oxidized by complex IV, it is also expected that cytochrome c will primarily be in reduced form [38], thus $r(t) \to c_0$ as $t \to \infty$.

The effect of changing initial levels of cytochrome c reduced, or r, is shown in Figs. 3.6 and 3.7. Setting $r(0) = 50$ nmol/mL resulted in similar output to that of normal conditions, with a minor increase in overall activity, shown by the slightly more rapid equilibration and full consumption of oxygen compared to baseline. Also, since increasing initial cytochrome c concentration affects total cytochrome c concentration c_0, $r(t)$ shows a higher final concentration since $r(t) \to c_0$ as $t \to \infty$.

Simulating low initial r concentration by setting $r(0) = 5.0 \times 10^{-5}$ nmol/mL yields output matching that of the baseline initial conditions, as shown below in Fig. 3.7. Oxygen is consumed at a similar rate compared to normal conditions until oxygen is fully consumed at approximately $t = 450$ s.

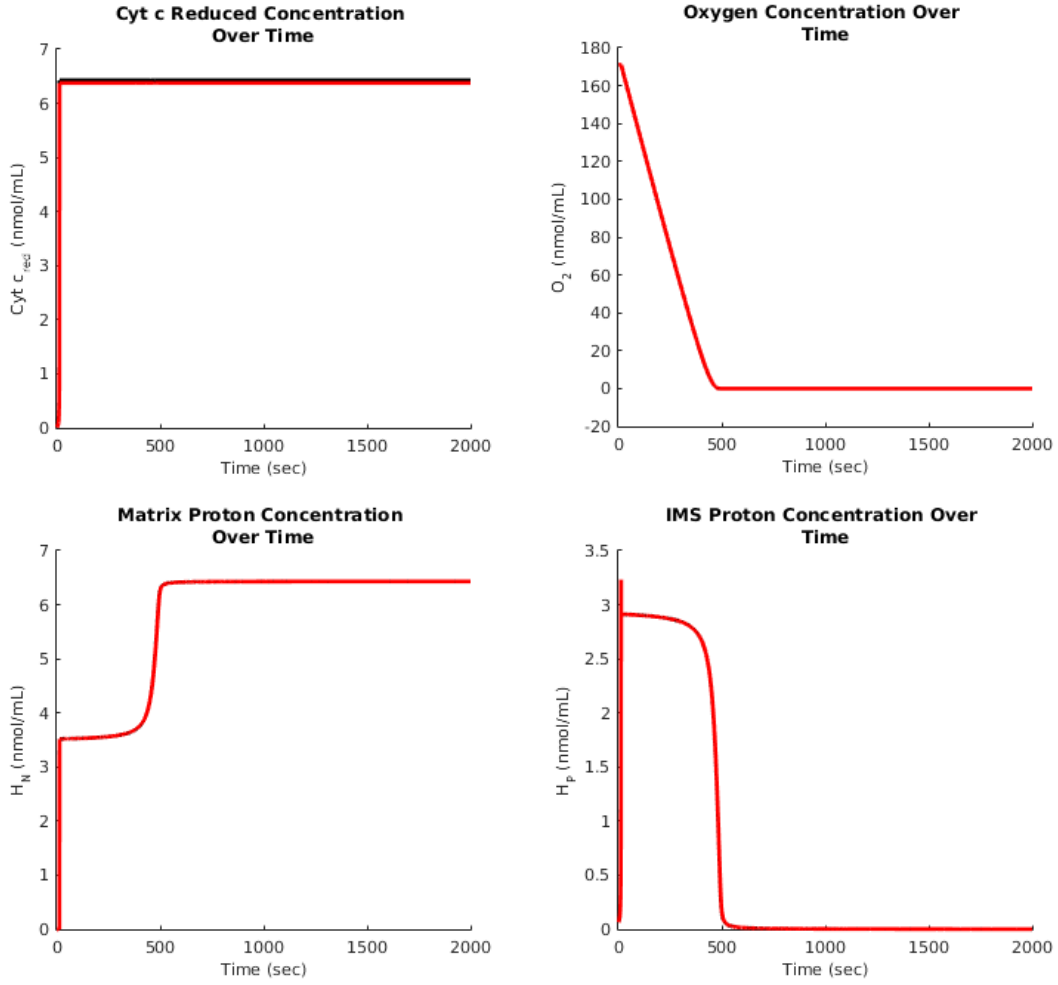
When the initial oxygen concentration is increased to $o(0) = 1.71055 \times 10^5$ nmol/mL, it is expected that cytochrome c reduced levels remain fairly consistent or consumed slowly as oxygen is quickly consumed. Equation (2.6) illustrates that increasing oxygen would also greatly increase oxygen consumption rate. Fig. 3.8 shows constant consumption of oxygen, however, which is due to the limitation of the low level of initial cytochrome c reduced in the system since $r(0) = 0.0109$ nmol/mL. Cytochrome c reduced is continually consumed by complex IV

Figure 3.6: Model output with a high initial concentration of cytochrome c reduced. Here, the initial concentration of cytochrome c reduced is set to $r(0) = 50$ nmol/mL. Black lines represent output under normal conditions, whereas red lines represent output under altered initial conditions.



at a maximal rate, constrained by the amount of cytochrome c reduced produced by complexes I-III, evident in the graph depicting OCR over time. This maximal OCR affects proton balance, since the activity of the ETC will continually translocate protons from the matrix, resulting in ω remaining at a low final concentration ω_f. Maximal activity of the ETC will force protons against the *pmf* faster than complex V can normally translocate protons into the matrix, thus resulting in an increase in ρ. Equilibrium has not been reached on the timescale simulated, however, due to the abundance of oxygen available to the ETC. Validating this output of the model may be necessary with additional experimental data.

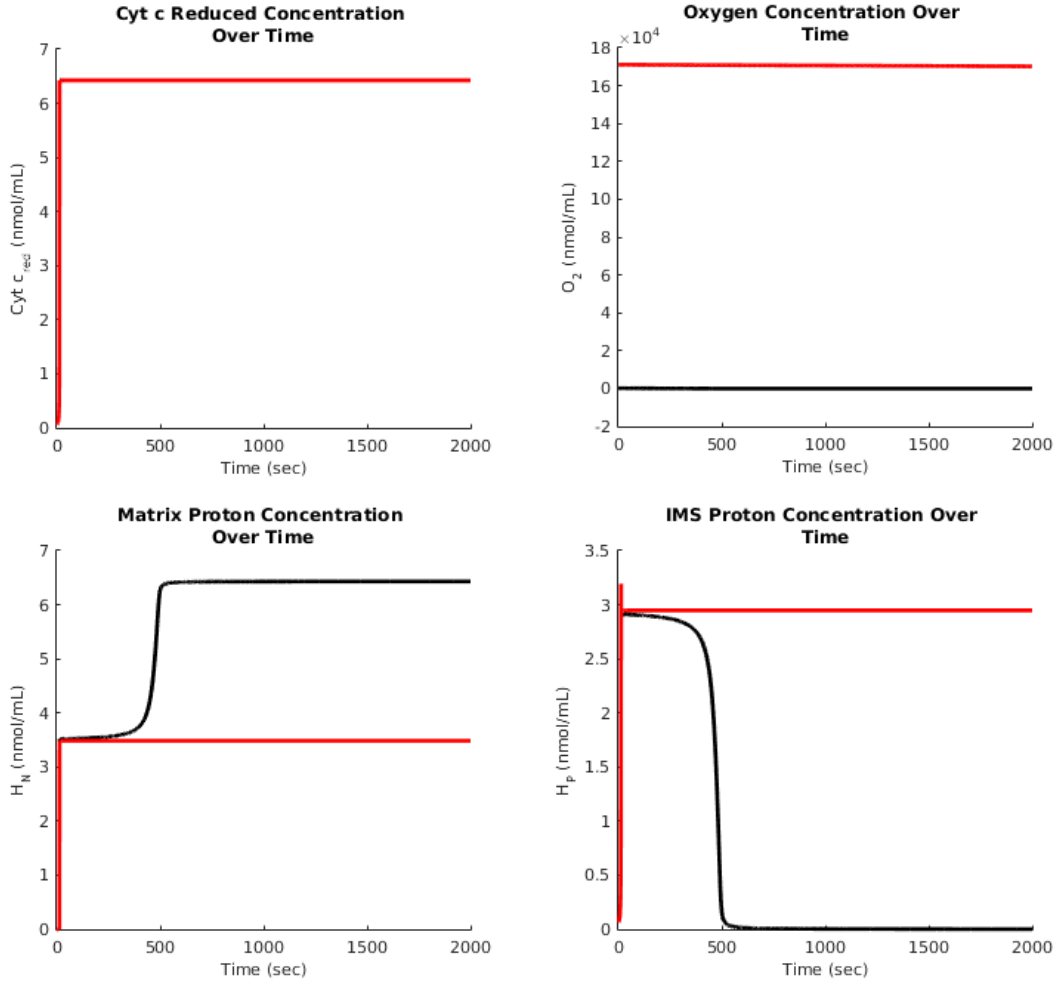
Figure 3.7: Model output with a low initial concentration of cytochrome c reduced. Here, the intiial concentration of cytochrome c reduced is set to $r(0) = 5.0 \times 10^{-5}$ nmol/mL. Black lines represent output under normal conditions, whereas red lines represent output under altered initial conditions.



When oxygen is set to a small initial concentration $o(0) = 0.171055$ nmol/mL, oxygen is assumed to fully deplete, thus $o(t) \to 0$ nmol/mL as $t \to \infty$. Due to the small amount of oxygen available to the system, it is fully consumed rapidly. Full consumption of oxygen results in the cessation of the entire chain, which results in rapid equilibration of ω, ρ and r. Since this results in similar activity of the system under normal conditions, but reaching a state of oxygen depletion much sooner, ω_f, r and ρ show similar steady state values.

Increasing proton concentration in the mitochondrial matrix resulted in rapid relaxation toward equilibrium, as shown in Fig. 3.10. Biologically, when the initial concentration of matrix protons

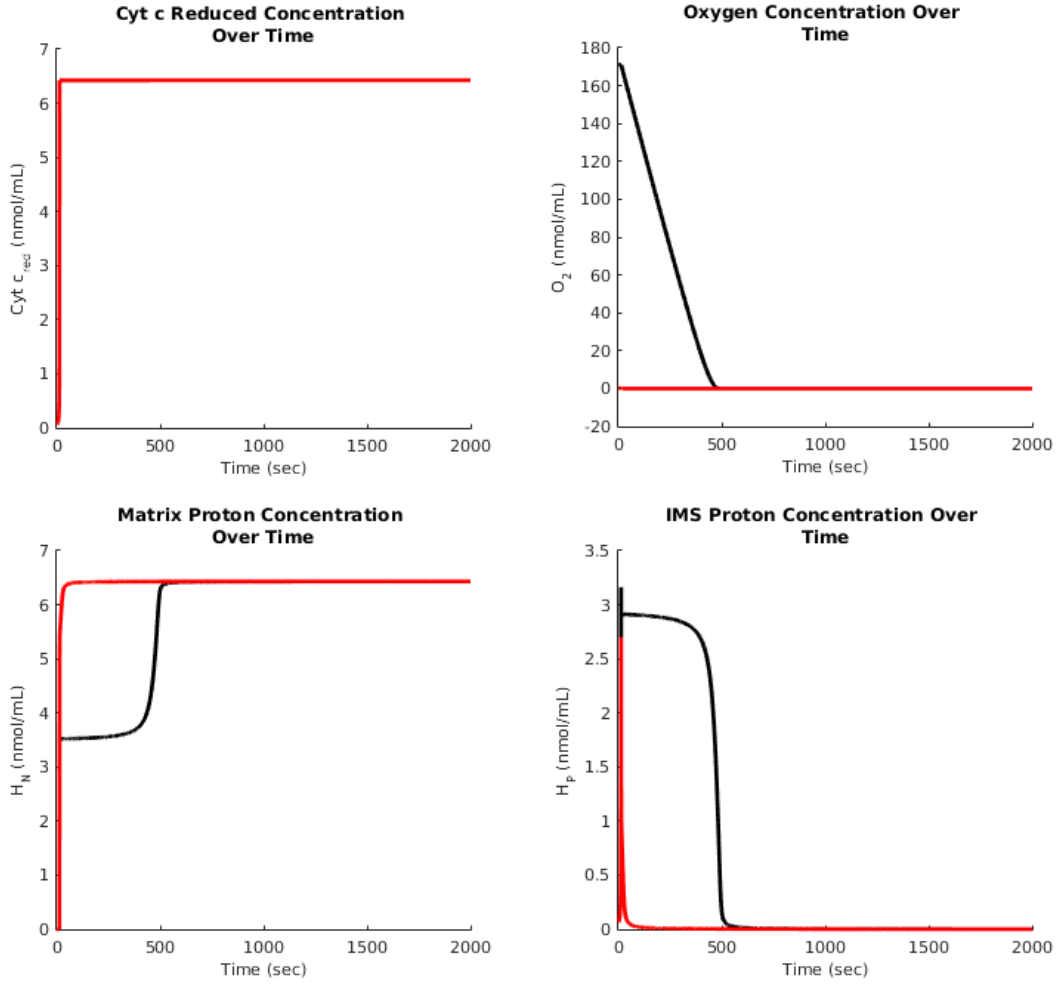
Figure 3.8: Model output with a high initial concentration of oxygen. Here, the initial concentration of oxygen is set to $o(0) = 1.71055 \times 10^5$ nmol/mL. Black lines represent output under normal conditions, whereas red lines represent output under altered initial conditions.



is increased, the proton gradient multiplicative factor $\frac{\omega}{\rho}$ is increased, resulting in rapid activity of the system driven by the *pmf* to equilibrate proton levels across the inner mitochondrial membrane. This matches the rapid initial activity shown in Fig. 3.10. It is also shown that $\omega(t)$ equilibrates to a higher ω_f than under normal conditions, which may be due to the higher initial concentration of overall protons available to the system.

Fig. 3.11 shows that a low starting concentration of matrix protons results in output matching that of normal conditions with a slight delay. A lowered concentration of initial matrix protons results in a smaller multiplicative factor $\frac{\omega}{\rho}$. This thereby reduces the initial rate of activity for

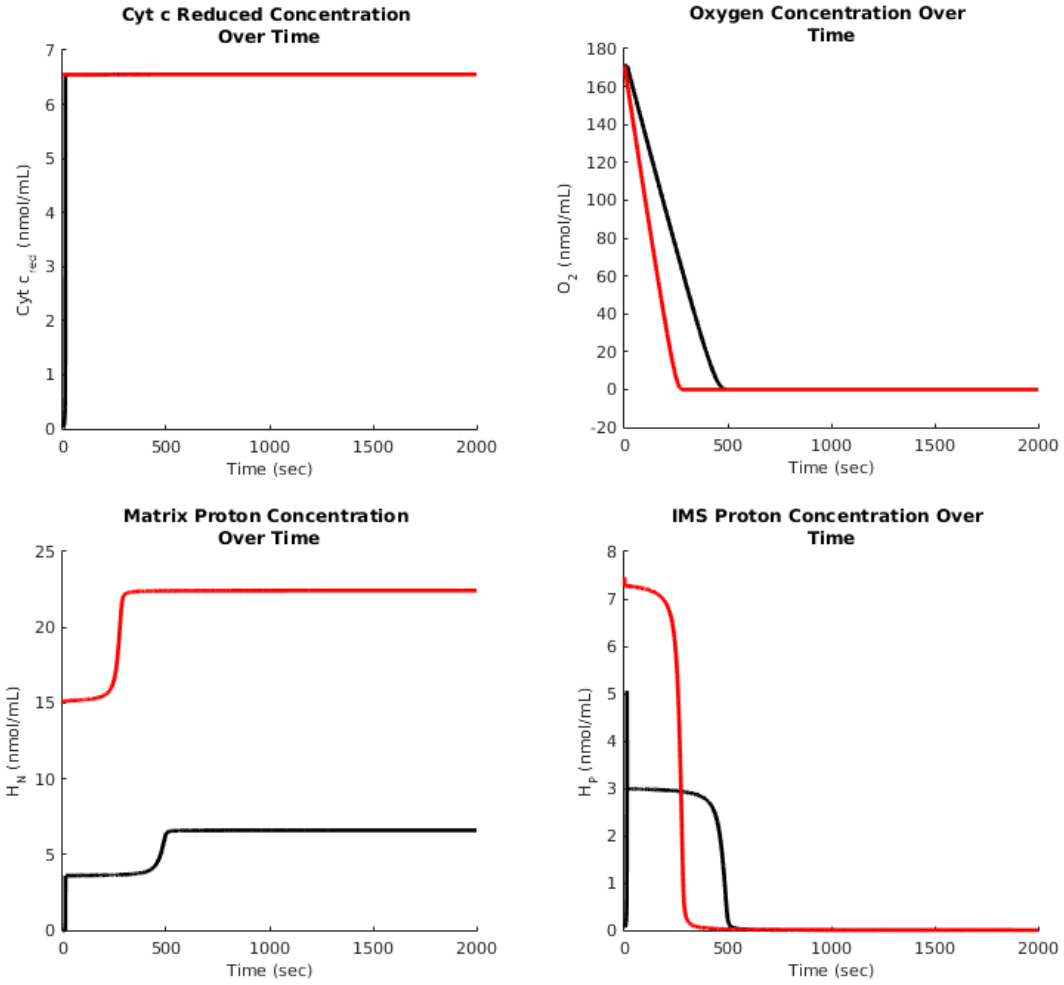
Figure 3.9: Model output with a low initial concentration of oxygen. Here, the initial concentration of oxygen is set to $o(0) = 0.171055$ nmol/mL. Black lines represent output under normal conditions, whereas red lines represent output under altered initial conditions.



F_0. Due to the small concentrations of protons, however, the ratio $\frac{\omega}{\rho}$ quickly approaches values similar to those under normal conditions. This can explain the slight delay in functioning under low matrix proton conditions.

Increasing IMS proton concentration shows a similar effect to increasing matrix proton concentration, as shown below in Fig. 3.12. This may be occurring since higher ρ results in rapid activity of complex V, which drives protons inward to the matrix. This rapidly increases ω, which then results in similar general output to that of increased initial ω. Fig. 3.13 shows that

Figure 3.10: Model output with a high initial concentration of matrix protons. Here, the initial concentration of matrix protons is set to $\omega(0) = 15.85$ nmol/mL. Black lines represent output under normal conditions, whereas red lines represent output under altered initial conditions.



decreasing initial proton concentrations in both IMS and the matrix results in a short delay of the function of the system.

Investigating the effect of changing initial concentrations of the state variables of the model provides insight into the flexibility of the model in simulating various biological environments. The output shown in Figs. 3.6 - 3.13 shows that the model can sufficiently predict mitochondrial functioning after altering the initial conditions of the model.

Figure 3.11: Model output with a low initial concentration of matrix protons. Here, the initial concentration of matrix protons is set to $\omega(0) = 1.585 \times 10^{-5}$ nmol/mL. Black lines represent output under normal conditions, whereas red lines represent output under altered initial conditions.

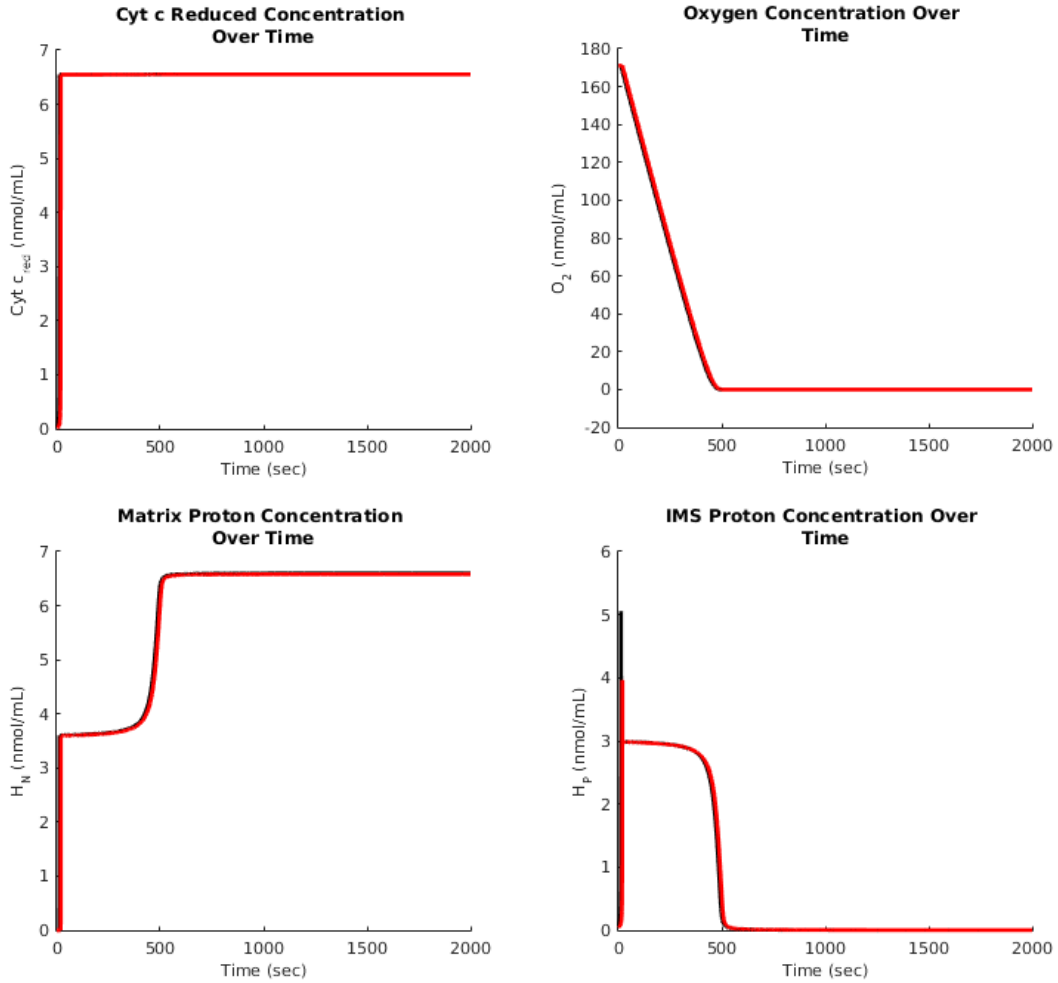


Figure 3.12: Model output with a high initial concentration of intermembrane space protons. Here, the initial concentration of intermembrane space protons is set to $\rho(0) = 39.81$ nmol/mL. Black lines represent output under normal conditions, whereas red lines represent output under altered initial conditions.

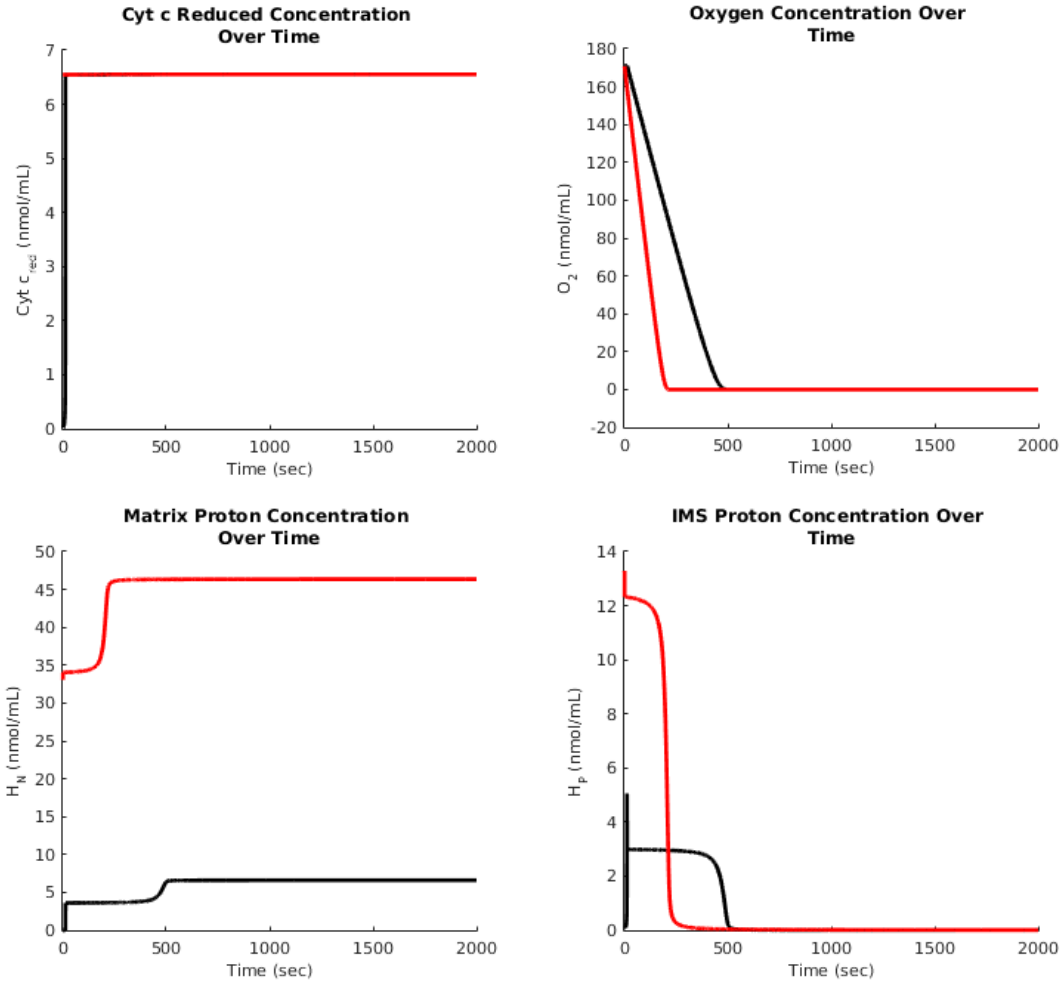
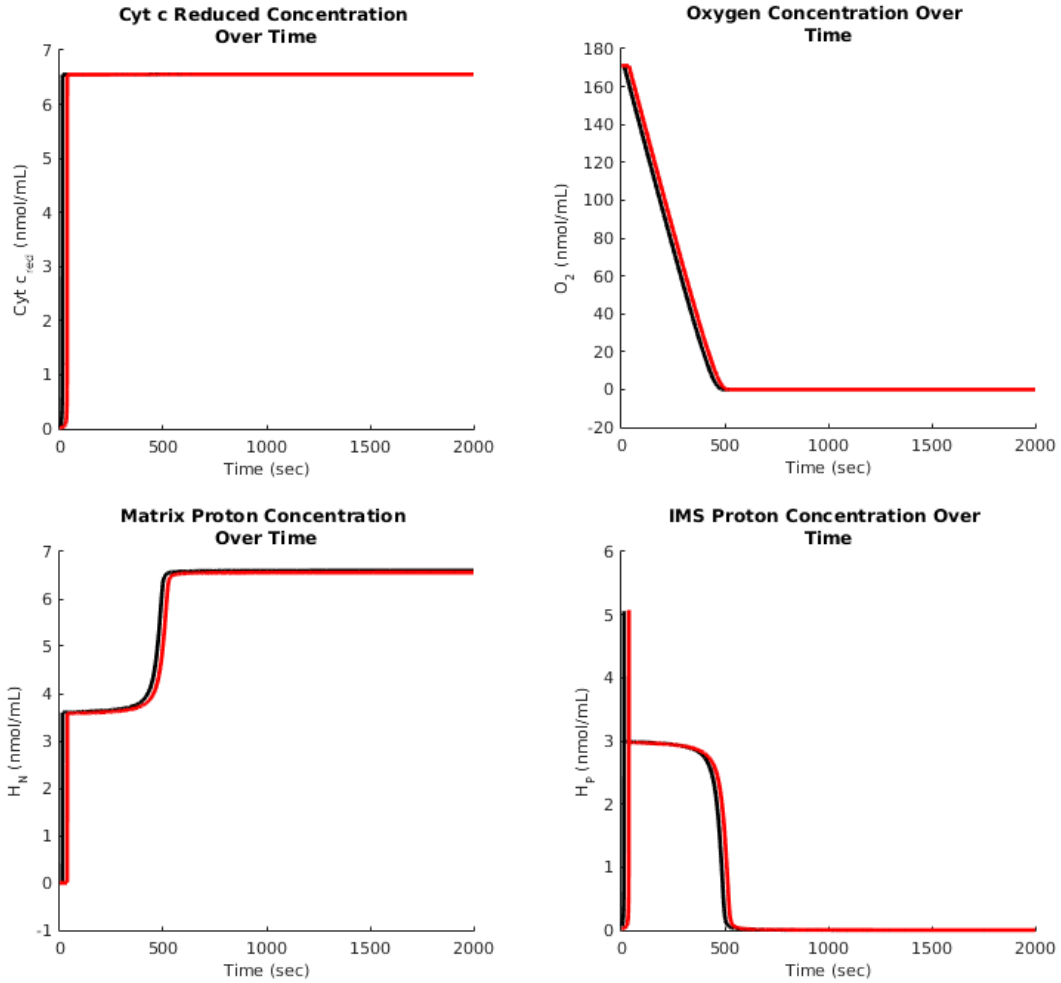


Figure 3.13: Model output with a low initial concentration of intermembrane space protons. Here, the initial concentration of intermembrane space protons is set to $\rho(0) = 3.981 \times 10^{-5}$ nmol/mL. Black lines represent output under normal conditions, whereas red lines represent output under altered initial conditions.



3.2.2.1 Comparing Oxygen Consumption Rate

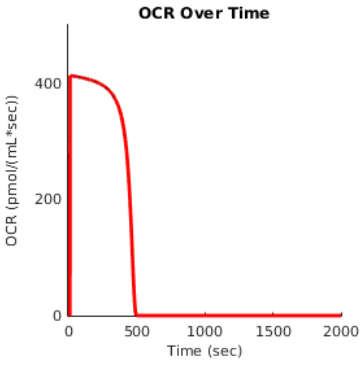
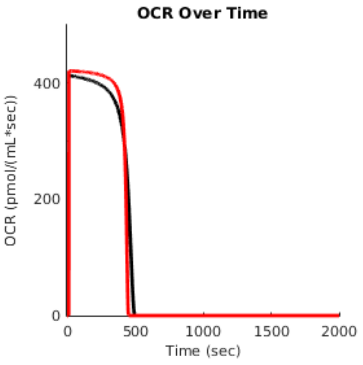
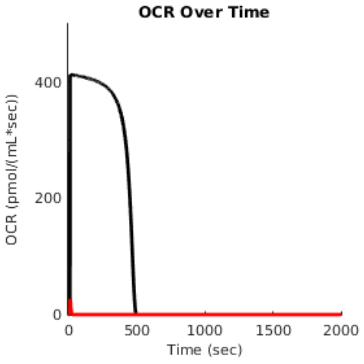
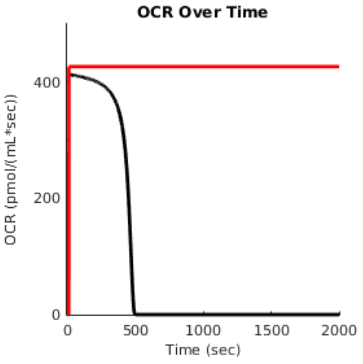
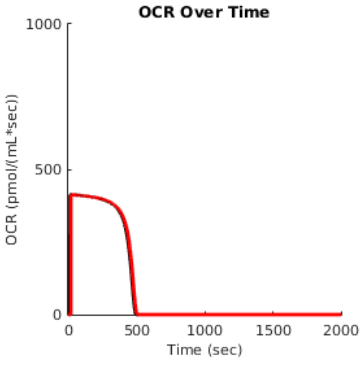
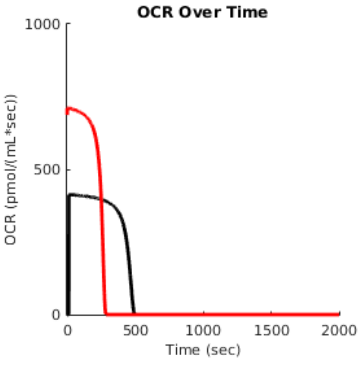
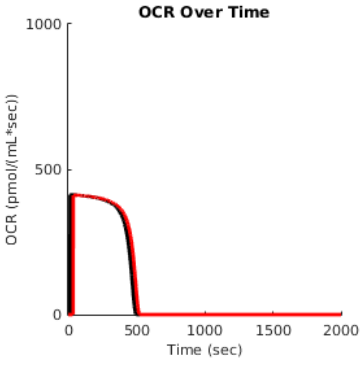
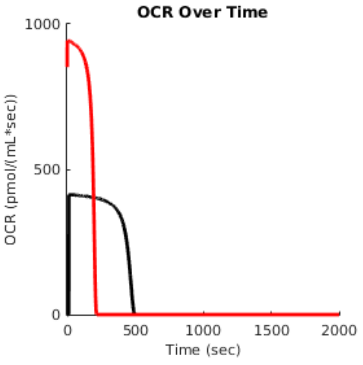
OCR provides an important measure of mitochondrial activity, and in particular activity of complex IV. OCR is often used to quantify mitochondrial activity in an experimental setting [39][40][41]. Determining the effect of changing initial concentrations of the model on OCR is thus an important consideration for the applicability of the model. Output OCR curves for the conditions tested above have been aggregated into Table 3.3 below, juxtaposed for comparison purposes. In each OCR graph shown, the black lines represent unaltered initial conditions, whereas the red lines show altered initial conditions.

Table 3.3 shows that both increased and lowered initial concentrations of cytochrome c reduced results in similar activity to that of normal complex IV behaviour. This indicates that initial cytochrome c reduced concentration is not a significant contributor to the output of the model, which is also supported by the sensitivity analysis shown in Section 3.2.3.

As expected, OCR is particularly sensitive to the starting concentration of oxygen. An abundance of oxygen results in a high, constant rate of consumption, whereas an absence of oxygen produces no consumption.

Changing initial concentrations of matrix and IMS protons similarly affect the shape of the OCR curve over time. High initial proton concentrations result in an initial rapid consumption of oxygen, followed by a sharp decrease in OCR. OCR continues to decrease slowly until full oxygen depletion. These results indicate that higher initial concentrations of protons result in variable OCR rates prior to oxygen depletion, which may be a result of higher proton flux rates resulting in higher overall activity of the ETC. Lower initial concentrations result in normal behaviour, which may be due to a rapid equilibration of proton concentrations toward normal conditions.

Table 3.3: Summary of the effect of changing initial conditions OCR. The figures displayed illustrate the effect of each substrate on output OCR, where oxygen shows the greatest control over OCR.

Substrate Changed	**Low Concentration Condition**	**High Concentration Condition**
Cytochrome C Reduced		
Oxygen		
Matrix Protons		
IMS Protons		

3.2.3 Sensitivity Analysis

After looking at the effect of altering initial conditions on the response of the model, the effects of altering model parameters were also determined by using the sensitivity analysis outlined in Section 2.2.7.2. The sensitivity analysis used here quantifies the effect of altering parameters from their optimal values on the model response. The model response is evaluated through the error between $o(t)$ and experimental oxygen concentration data, as shown in Equation (2.17). Thus, the sensitivity of the model to each parameter specifically regards the sensitivity of oxygen concentration $o(t)$ to varying parameter values from the optimal set of values $\vec{p}^*$.

The results of sensitivity analysis, determined using Equation (2.18), are listed below in Table 3.4. Sensitivities to each parameter were determined for each biological condition separately; *baseline*, *oligomycin*, *FCCP* and *rotenone/antimycin A* conditions.

Table 3.4: Sensitivity values for the parameters of the model. These sensitivity values were evaluated separately for each subsystem in the model. Sensitivity values in bold are relatively large in contrast to other parameter in the same subsystem. The oligomycin and inhibited subsystems show little sensitivity to the parameters of the model.

Parameter	Baseline	Oligomycin	FCCP	Inhibited System
$V_{max_{cI\text{-}III}}$	0.053	0.0005	0.003	-
$K_{m_{cI\text{-}III}}$	0.032	0.0005	0.003	-
$V_{max_{cIV}}$	**15.63**	0.0011	**261.6**	0.162
$K_{m_{cIV}}$	0.770	0.0001	5.110	0.028
K_{cIV}	0.760	0.0001	4.996	0.028
$V_{max_{cV}}$	**14.19**	-	-	-
$K_{m_{cV}}$	7.963	-	-	-
K_{cV}	5.440	-	-	-
D_h	-	-	0.081	0.001
Cyt c_{red}	0.072	0.0009	0.006	0.000

Sensitivity values indicate the relative effect of altering each parameter on output oxygen concentration of the model; the larger the sensitivity values the greater the effect. Given equation (2.18), sensitivity values are relative to the best fit error value and are therefore unitless.

The baseline subsystem shows particular sensitivity to the parameters $V_{max_{cIV}}$ and $V_{max_{cV}}$. The FCCP subsystem shows particular sensitivity to the parameter $V_{max_{cIV}}$. Also, the oligomycin and inhibited systems both show little sensitivity to model parameters, due to the slowed activity

of the system in both of these conditions. Interpretations of these sensitivity values are discussed further in Section 4.2.1.

3.3 Model Interface

The simulation model was coded in MATLAB R2014a and MATLAB R2015a (Mathworks). The model was programmed using a functional approach, thus using several *.m* files to collectively simulate mitochondrial function. The content of each *.m* file is located in Appendix C, illustrating the functionality of all MATLAB files developed for this project: *main.m*, *setup.m*, *finalgui.m*, *baselineSystem.m*, *oligoSystem.m*, *fccpSystem.m*, *inhibitSystem.m*, *data_formatter.m*, *solver.m*, *fitness.m*, *sensitivityAnalysis.m*, *sensitivitySolver.m*, and *analyzeResults.m*

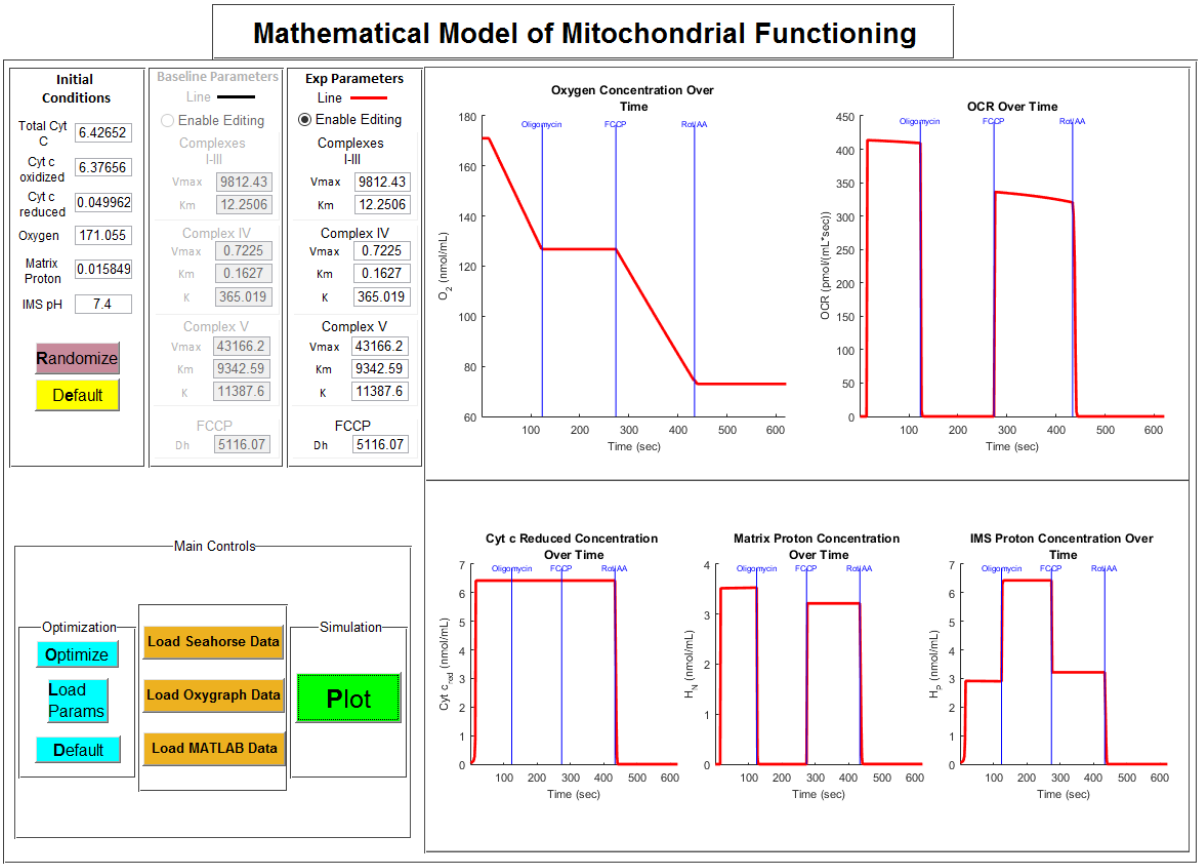
All MATLAB code was developed fully for the purposes of this project. The graphical user interface (GUI) shown in Fig. 3.14 was initially created using the built-in GUI development tool *GUIDE*. Functionality of the components and operation of the GUI was then expanded by writing specific functions in *finalgui.m*.

The functions *baselineSystem.m*, *oligoSystem.m*, *fccpSystem.m* and *inhibitSystem.m* each contain the equations for each biological section as detailed above in Section 2.2.6. The *solver.m* function implements the differential equation solver *ode45* to integrate the appropriate equations over all segments of the model as outlined in Section 2.2.7. The *data_formatter.m* function reads experimental data from an Excel file and imports it, along with time-stamped labels, into MATLAB for use in the model. The *fitness.m* function calculates the objective function as defined in Section 2.2.7 for optimization. The function *sensitivitySolver.m* carries out integration of the model for each biological condition separately for use in the sensitivity analysis outlined below in Section 2.2.7.2 and carried out in the code *sensitivityAnalysis.m*.

The primary code for running the model was programmed in *main.m*. This function calls *setup.m* which initializes all parameters, conditions, and data structures used in the model, and then opens *finalgui.m* to allow interactive control of the model.

This *finalgui.m* function handles all user activities, such as manipulating the parameter or initial values of the model. As illustrated in Fig. 3.14, the GUI allows for modification of the initial

Figure 3.14: The graphical user interface of the simulation model. This figure shows the developed graphical user interface (GUI) of the simulation model. This GUI was coded in MATLAB R2014a, and simulates both experimental and baseline simulations of the model in parallel. The operator of the GUI can edit the parameters and initial conditions of each simulation using the controls shown, and the output of each simulation is shown after the *Plot* button is clicked.



conditions of the simulation as well as the values of the parameters for both baseline and comparison, or experimental, simulations. Both baseline and experimental simulations are carried out in parallel, to allow for direct comparison between model outputs. The parameters for the baseline condition are fixed by default, whereas the parameters for the experimental condition are open to modification by the user. There are also functions that have been incorporated into the GUI for generating randomized sets of initial conditions or parameter sets for exploring various biological conditions. The GUI also allows the user to: save progress, load progress, save or resize output graphs, collectively save the output of the model to a single image, and save a screen shot of the model output and modifications to the parameters of the model producing

that output in a single image. Instructions for operating the functions of the GUI are found in Appendix B.

Chapter 4

Discussion

4.1 Biological Considerations

In the initial oxygen concentration data acquired by the Seahorse XF24 Analyzer, shown in Fig. 3.2, it appears that oxygen is periodically supplied to the system throughout the experiment. This is not the case, however. When oxygen is measured over a small interval, the measurement plunger measures this in a small closed chamber of approximately 7 μL by moving downward in the well. After a series of measurements, the plunger then lifts and results in the entire well mixing. This also allows the level of oxygen to return to the baseline oxygen level available in the entire well. This allows the XF24 Analyzer to determine the rate of oxygen consumption over an interval of time and ensure that oxygen will be available for the next interval of measurements. So although OCR measurements are accurate, the overall behavior of oxygen is influenced by this process of mixing. Quantitatively, this mixing process thus represents periodic increases in oxygen concentration into the system, namely some anonymous input function $\pi(t)$ into $\frac{do}{dt}$. This would require introducing a function specific to the mechanical operation of the Seahorse, however, not to the functioning of the mitochondria. Thus, the need for the Oroboros Oxygraph-2k.

Oxygen concentration from the Oxygraph-2k is continuous and accurately represents direct measurements of oxygen concentration over time without interruption. To validate that the oxygen

levels are consumed at proper rates, the average rate of consumption within each biological condition in the Oxygraph-2k data was normalized and compared to that of the Seahorse XF24 data. As evident in contrasting Fig. 3.3 and the red OCR curve shown in Fig. 3.1, the pattern of consumption rates in the Oxygraph-2k data matched that of the Seahorse XF24 Analyzer data, validating the response of the mitochondria to each biological reagent qualitatively.

In addition to the issue of using Seahorse XF24 Analyzer data for calibration, there is also a concern for approximating pH by IMS protons $\rho(t)$, as briefly described in Section 3.2.2. In modelling the system at solely the level of the electron transport chain, the external concentration of protons, which is thus the concentration of IMS protons, can approximate the pH of the system [1]. This was used to determine the initial concentration $\rho(0)$ by determining the concentration of $\rho(t)$ that corresponded to $pH = 7.4$, which was the pH measured by the Seahorse XF24 Analyzer. The issue in making this assumption is that the measurements of pH carried out by the Seahorse XF24 Analyzer are made at a cellular level, which may not fully represent the environment in the mitochondria, specifically the IMS. To reduce reliance on this assumption, measurements of proton concentration on both sides of the inner mitochondrial membrane will be required. Given the aim of the model, however, this assumption provides an adequate approximation for the initial concentration for $\rho(t)$. Assuming a resting pH gradient of -0.4 units, within the bounds of previous determinations [35][36][37], allows this pH measurement to also determine an initial concentration for $\omega(t)$. Thus direct measurement of the pH gradient on the inner mitochondrial membrane will allow for a more precise determination of $\omega(0)$.

Investigating the concentration of cytochrome c reduced in the calibrated model, as shown in Fig. 3.5, shows that the oxidized form of cytochrome c is dominant at $t = 0$ s. Quickly, cytochrome c becomes reduced until the reduced form dominates, where cytochrome c reaches a reduction proportion of 85 %. The reduction proportion achieved during baseline is higher than in previous studies, such as the 15 - 30 % reduction proportion shown by Muraoka and Slater (1969) [38]. This over-reduction of cytochrome c may simply be due differences in the experimental protocols used, or due to the reliance of the system on cytochrome c oxidized as input into the ETC, instead of the carrier NADH, which was assumed to be in abundance. This can be further investigated upon expansion of the model, as described in the next section.

Once cytochrome c reaches a stable reduction proportion, here achieved at 85 % in the baseline subsystem, reduction will no longer proceed at a significant rate. Although injection of oligomycin is expected to alter this reduction proportion, it appears in Fig. 3.5 that this does not occur to any visible extent. Thus, expansion of the model may be required to further replicate proper cytochrome c dynamics. This can be addressed by describing in detail complexes I-III independently, allowing for a refinement of cytochrome c dynamics.

4.2 Mathematical Considerations

4.2.1 Sensitivity Analysis

The results of the sensitivity analysis are listed above in Table 3.4. This analysis showed that the model is most sensitive to changes in model parameters $V_{max_{cIV}}$ and $V_{max_{cV}}$ in the baseline subsystem; and $V_{max_{cIV}}$ in the FCCP subsystem. These results are interpreted here using the underlying biology of the system.

The oligomycin and inhibited subsystems show significant decreases in sensitivity values to all parameter values. This is a consequence of the drop in mitochondrial activity under these conditions.

In the baseline subsystem, the model shows sensitivity to changes in the parameters $V_{max_{cV}}$ and $V_{max_{cV}}$. Sensitivity to $V_{max_{cIV}}$ reflects the control of this parameter on the maximal activity of complex IV, which affects the rate at which oxygen consumption can proceed. This thereby directly affects the accuracy of the model's output compared to calibration data. Sensitivity to this parameter is also exhibited in the FCCP subsystem, where the same argument holds. Sensitivity to $V_{max_{cV}}$, however, illustrates the importance of complex V to the operation of the ETC, in that it acts as the only modelled method of dissipating the generated pmf. In the FCCP condition, the means of pmf dissipation is accomplished through F_6, controlled by the parameter D_h. The model does not prove to be sensitive to this parameter, which may be due to F_6 modelling a passive process of proton movement rather than a tightly controlled process such as in F_5. This can be further investigated once the model is expanded and additional data regarding measurable proton concentrations are incorporated.

4.2.2 Expanding the Model

As discussed in section 2.2.1, complex IV became the main focus of the model since experimental data was inherently collected directly from complex IV activity. This resulted in the description of complexes I-III by a single function, F_0. Further collection of experimental data measuring continuous concentration levels of other substrates in the system, such as NADH, cytochrome c reduced, and/or proton levels, will allow model development into a more complete and robust representation of the ETC without representing complexes I-III by F_0.

To extend the current model, kinetic descriptions of all biochemical reactions in the ETC must be determined. The chemical reactions to be modelled were described above in section 1.1.2, and are again listed here for clarity:

$$complex\ I : NADH_2 + 4H_N^+ + Q \longrightarrow NAD^+ + QH_2 + 4H_P^+ \tag{4.1a}$$

$$complex\ II : Succinate + 2H_N^+ + Q \longrightarrow Fumarate + QH_2 \tag{4.1b}$$

$$complex\ II_{reverse} : Fumarate + MQH_2 \longrightarrow Succinate + MQ + 2H_N^+ \tag{4.1c}$$

$$complex\ III : QH_2 + 2Cytc_{ox} + 2H_N^+ \longrightarrow Q + 2Cytc_{red} + 4H_P^+ \tag{4.1d}$$

$$complex\ IV : 2Cytc_{red} + 4H_N^+ + \frac{1}{2}O_2 \longrightarrow 2Cytc_{ox} + H_2O + 2H_P^+ \tag{4.1e}$$

Notice that since complex II also operates reversibly, this reaction was also included in the reaction scheme for the ETC, although it is not an energetically favourable reaction and thus not occur often.

To determine the appropriate kinetic equations for each biochemical reaction in equations (4.1), the mechanism underlying their function must be analyzed. Given the parallels between each reaction, it appears that each complex operates to carry out a redox reaction between *two* substrates. It is not entirely known, however, whether both substrates can freely bind to the relevant complex or whether a single substrate (or *coenzyme*) must first be bound before a substrate can be bound. Thus, there exists two possible kinetic mechanisms, *activation kinetics* or *two-substrate kinetics*.

Activation kinetics requires a specific substance, referred to as an *activator*, to firstly bind to the complex before a substrate may bind. Complexes I, II, III and IV may all follow activation

kinetics, with the coenzyme involved in each complex acting as the activator. In complex I, for example, NADH is an active coenzyme in the redox reaction carried out by complex I, thus NADH would be the activator and Q would be the substrate operated upon. Identification of activator and substrates for each complex is listed in Table 4.2. Two-substrate kinetics involves the ability of both substrates to freely bind to the complex, but the reaction only proceeds once both substrates are bound. Only complexes I, II and III fit with this kinetic mechanism. This is invalid for complex IV since two electrons are passed to complex IV by 2 molecules of cytochrome *c*, it is only once these electrons reach the Cu-Fe bi-center at the end of the redox chain of complex IV can O_2 bind [1]. Thus, cytochrome *c* must activate complex IV before O_2 can bind, disallowing the two-substrate kinetic scheme.

Complexes I-III can thus be modelled as either having activation or two-substrate kinetics, except for complex IV. Thus, there are $2^3 = 8$ different model types, which is summarized below in Table 4.1.

Table 4.1: The set of all possible kinetic model types for a full description of the ETC. Complexes I, II, and III can each be described by using either activation or two-substrate kinetics, whereas complex IV is solely described by activation kinetics. This results in eight possible model types when describing the entire ETC.

Complex I	**Complex II**	**Complex III**	**Complex IV**	**Model Type**
Activation	Activation	Activation	Activation	1
Activation	Activation	Two-Substrate	Activation	2
Activation	Two-Substrate	Activation	Activation	3
Activation	Two-Substrate	Two-Substrate	Activation	4
Two-Substrate	Activation	Activation	Activation	5
Two-Substrate	Activation	Two-Substrate	Activation	6
Two-Substrate	Two-Substrate	Activation	Activation	7
Two-Substrate	Two-Substrate	Two-Substrate	Activation	8

The kinetics rate equation for each type of kinetic model have been derived and detailed in Appendix A, where the rate equation for activation kinetics is:

$$v = \frac{V_{max} S}{K_M\left(1 + \frac{K_1}{A}\right) + S} \tag{4.2}$$

where A is the amount of activator present, and S is the amount of substrate. The rate equation for two-substrate kinetics is:

$$v = \frac{V_{max}AB}{\frac{1}{K_aK_b} + \frac{A}{K_b} + \frac{B}{K_a} + AB}. \tag{4.3}$$

where A is the amount of one substrate and B is the amount of another substrate.

By separating the biochemical reactions in equation (4.1) by reagent, an equation for the time evolution of each reagent can be determined. By letting F_i represent the speed of the reaction for each i-th complex, where $i = \{1, 2, 3, 4, 5\}$, the activity of each complex in the ETC is represented by some function F_i. With these functional representations, the rate of production for all compounds in the system can be modelled by using the techniques illustrated in Appendix A, yielding:

$$\frac{d[NADH]}{dt} = -F_1(NADH, Q) \tag{4.4a}$$

$$\frac{d[Q]}{dt} = -F_1(NADH, Q) - F_2(Succinate, Q) + F_3(QH_2, Cytc_{ox}) \tag{4.4b}$$

$$\frac{d[NAD^+]}{dt} = F_1(NADH, Q) \tag{4.4c}$$

$$\frac{d[QH_2]}{dt} = F_1(NADH, Q) + F_2(Succinate, Q) - F_3(QH_2, Cytc_{ox}) \tag{4.4d}$$

$$\frac{d[succinate]}{dt} = -F_2(Succinate, Q) + F_{2\mathrm{rev}}(Fumarate, MQH_2) \tag{4.4e}$$

$$\frac{d[fumarate]}{dt} = F_2(Succinate, Q) - F_{2\mathrm{rev}}(Fumarate, MQH_2) \tag{4.4f}$$

$$\frac{d[Cytc_{ox}]}{dt} = -2F_3(QH_2, Cytc_{ox}) + 2F_4(Cytc_{red}, O_2) \tag{4.4g}$$

$$\frac{d[Cytc_{red}]}{dt} = 2F_3(QH_2, Cytc_{ox}) - 2F_4(Cytc_{red}, O_2) \tag{4.4h}$$

$$\frac{d[O_2]}{dt} = -\frac{1}{2}F_4(Cytc_{red}, O_2) \tag{4.4i}$$

$$\frac{d[H_2O]}{dt} = F_4(Cytc_{red}, O_2) \tag{4.4j}$$

$$\begin{aligned}\frac{d[H_N^+]}{dt} &= -4F_1(NADH, Q) - 2F_2(Succinate, Q) + 2F_{-2}(Fumarate, MQH_2) \\ &\quad - 2F_3(QH_2, Cytc_{ox}) - 4F_4(Cytc_{red}, O_2) + F_5(H_N^+, H_P^+) + F_6(H_N^+, H_P^+)\end{aligned} \tag{4.4k}$$

$$\begin{aligned}\frac{d[H_p^+]}{dt} &= 4F_1(NADH, Q) + 4F_3(QH_2, Cytc_{ox}) + 2F_4(Cytc_{red}, O_2) - F_5(H_N^+, H_P^+) \\ &\quad - F_6(H_N^+, H_P^+)\end{aligned} \tag{4.4l}$$

System (4.4) shows the complete description of the ETC, where F_1, F_2, and F_3 can either be determined by Equation (4.2) or (4.3), based on the assumed model type outlined in Table 4.1. In applying these rate equations to F_1, F_2, and F_3, Table 4.2 below identifies A, S, and B in equations (4.2) and (4.3) applied to each function. The functionals F_4, F_5 and F_6 remain as in the current model, given by Equations (2.6), (2.8) and (2.13), respectively. F_6 will be 0 until FCCP is introduced into the system, similar to the current model.

Table 4.2: Activators and substrates for each complex in complexes I-III. Activation kinetics requires the identification of an activator A and a substrate S, identified here for complexes I, II, and III. Similarly, two-substrate kinetics requires the identification of one substrate A and a second substrate B, which are identified here for complexes I, II, and III.

Function	**Activation**		**Two-Substrate**	
	Activator (A)	Substrate (S)	Substrate 1 (A)	Substrate 2 (B)
F_1	NADH	Q	NADH	Q
F_2	Succinate	Q	Succinate	Q
F_3	QH_2	Cyt c_{ox}	QH_2	Cyt c_{ox}

Extension to the entire ETC will require additional datasets to allow calibration for each component of the system. Potential experimental data that will allow for model extension includes measuring the concentration and consumption rates of: NADH, matrix and IMS protons, succinate, cytochrome c, and H_2O.

4.3 Implications of the Model

4.3.0.1 Utility of the Model

The model has three primary purposes in a research setting:

- Allowing the simulation and prediction of mitochondrial functioning under different conditions in a user-friendly environment,
- Generating, describing, and testing research hypotheses,

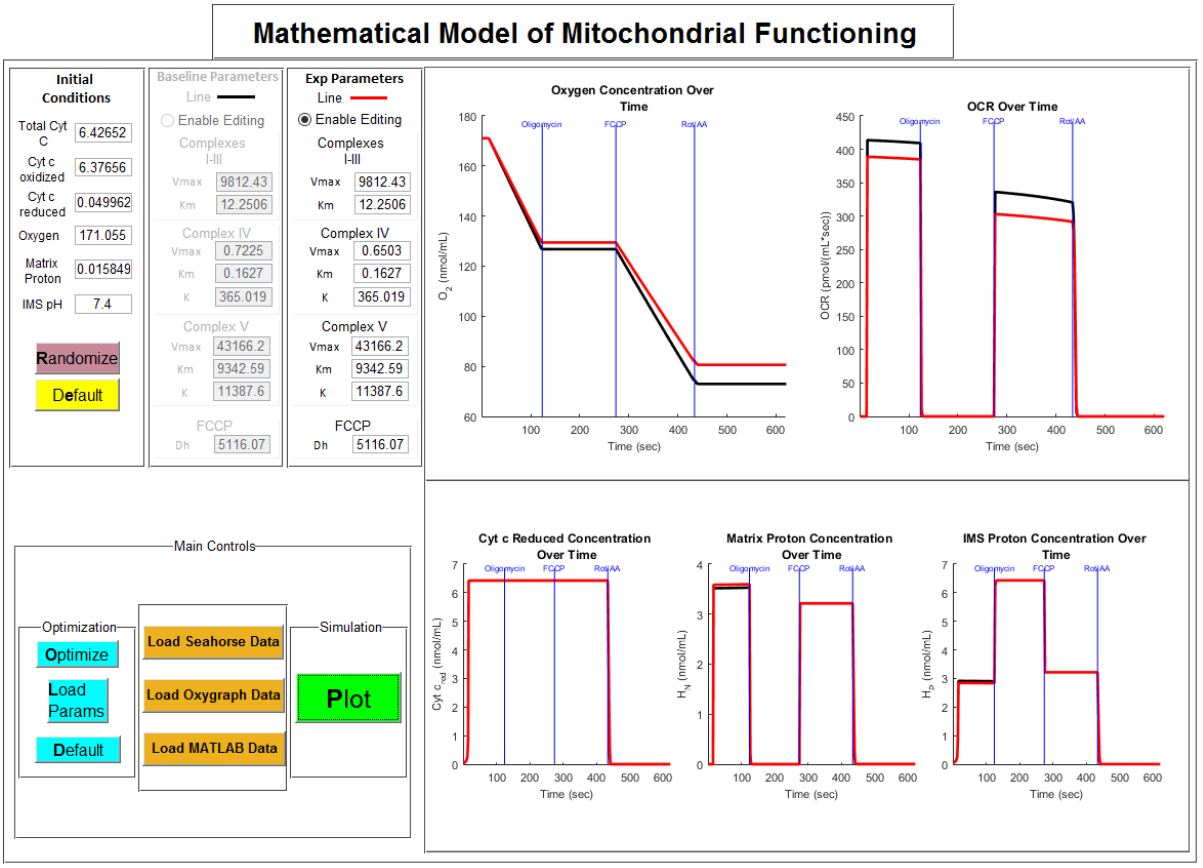
- Supplementing investigation of functional and dysfunction mitochondrial systems, particularly in disease states.

The first aim of the model has been accomplished through the development of a GUI, described in Section 3.3 and Appendix B, and shown in Fig. 3.14. The elements of the GUI have been made user-friendly by incorporating functions that allow for: flexible adaptation of the model, with the ability to re-simulate mitochondrial functioning after adaptation by clicking a single button; comparison between baseline and altered conditions by having two separate simulations of the model occurring in parallel; streamlined re-production of previous simulations, documentation of simulation output and conditions, and loading of additional data.

The second use for the model is thus naturally available to researchers using this GUI. The model can be manipulated and re-simulated directly, opening applicability to those that do not have familiarity with altering and running MATLAB scripts. The GUI has the flexibility for real-time modification and generating simulations of mitochondrial functioning under different conditions. This allows users to modify the model and investigate how these modifications may affect the system without biological experimentation. The results predicted by the model provide insight into possible behavior of mitochondria under the introduced modifications, which may assist in generating or investigating research hypotheses *in silico*. For example, Fig. 4.1 shows the effect of altering the maximal reaction rate of complex IV activity by lowering the $V_{max_{cIV}}$ parameter one order of magnitude, from 0.7225 to 0.6503. It appears that not only is oxygen consumption reduced, as expected, but the protons across the membrane also balance more slowly. The results of this simulation motivates investigating this effect further, illustrating the utility of the model in generating or motivating hypotheses and potential experimentation.

The results of the model may also be used to supplement biological experiments. The output of the model can corroborate experimental results when the proper modifications to the model's parameters or initial values are introduced. Re-calibrating the model to additional data sets, such as data collected from disease-state conditions, may also suggest potential mechanisms of dysfunction in the mitochondria. Upon re-calibration to these data sets, changes to the baseline parameter set may provide insight into specific changes in mitochondrial functioning.

Figure 4.1: Simulating the model after adjusting a model parameter. Using the model to simulate the effect of lowering $V_{max_{cIV}}$ by a factor of 10. This shows the output of simulating both baseline functioning, shown by the black lines, and altered functioning, shown by the red lines.



4.3.0.2 Next Directions

Although the model is still in its early stages of development, it is evident that the model itself is adaptable and can be easily expanded to incorporate different properties of the mitochondria, such as thermodynamic considerations. The model can additionally be further calibrated to additional data sets, particularly data from other state variables such as cytochrome c concentration, and proton levels on either side of the membrane, as discussed in Section 4.2.2.

The aim for the model is to be utilized as a research tool to investigate dysfunctional mitochondria in the central nervous system. Thus, additional data sets from functional mitochondria can also be accompanied by matched acquisition from disease-state mitochondria. This is a

direct application of the model, as outlined in 4.3.0.1. An interesting avenue for applying this model to disease-state mitochondria is to investigate the effect of Alzheimer's disease (AD) on mitochondrial function [4][42][43]. The experimental protocols carried out in this project can be repeated in either Alzheimer's-model mice, such as the 3xTg strain [44], or following application of amyloid-β protein to control mice [45]. Re-calibration of the model with these data will provide insight into potential mechanisms of alteration in the mitochondria, depending on which parameters of the model change and in what way.

Finally, the GUI can be also be developed further by introducing components that may be essential or helpful for usability. For instance, porting the model into a compiled language, such as $C++$, will allow for the creation of a standalone executable of the model. This will allow the model to be used without running the *main.m* script in MATLAB, expanding usability of the model beyond proprietary software.

Chapter 5

Conclusion

The model developed in this project has been shown to accurately represent oxygen concentration data acquired from mitochondrial samples. The current model thus provides an accurate description of complex IV activity under normal functioning, with the introduction of chemical perturbations to the mitochondria, and as well as in altered biological conditions.

The model was developed with a detailed, interactive graphical user interface (GUI) to provide ease-of-use in applying the model. The model can be applied to suggest potential avenues of investigation based on the simulated output. Re-calibration of the model to additional data sets can also suggest how different experimental conditions may lead to alterations in specific components of the system, based on how the model parameters must adapt to the additional data.

The model remains in its infancy, however, as there is still room for development and expansion. The approach was grounded in a wholly kinetic framework, which therefore requires expansion to incorporate thermodynamic considerations. Given the adaptability of the model, however, thermodynamic considerations can be introduced with appropriate additional data. Limited experimental data availability also constrained the model to focus on primarily modelling complex IV activity. Though, development and calibration of an extended model to additional sources of experimental data is the next step in expanding the model's framework. This will not only bolster the descriptive and predictive power of the model, but also expand avenues of utility.

Appendix A

Biochemical Kinetics Derivations

A.1 Kinetic Modelling of Chemical Systems

[1]

Biological functions are often the direct result of the interaction between various chemical species, resulting in some change in either the activity of one involved chemical or in the structure or function of a cellular component. In many of these cases, there are various complex interactions between a sequence of chemical reactions that ultimately result in some effect at the cellular level, and these are referred to as *pathways*. Pathways often incorporate the utilization of *enzymes*, protein units that *catalyze* or speed up and allow chemical reactions to proceed without themselves changing. In order to fully describe the activity of an enzymatic reaction, a description for a simple reaction is essential. Extending this general description then to enzymatic activity will allow for a method of deriving the reactions that may represent the function of the ETC enzymes.

[1]Sections A.1, A.1.1 and A.1.2 are summarized from [46] and [29] unless citations are otherwise indicated.

A.1.1 The law of mass action

A fundamental law of a chemical reaction is the *law of mass action*, which describes the rate at which chemicals react to form different chemical compounds. Take for example, the simplest form of a chemical reaction, involving reactants A and B producing some product C:

$$A + B \xrightarrow{k} C \tag{A.1}$$

Rate of product formation over time can then be expressed as $\frac{dC}{dt}$, which is dependent on the collision of reactants A and B and their probability of aggregation given a collision occurs. Thus the number of collisions per unit time is proportional to the amount of both A and B present in the system:

$$\frac{dC}{dt} = kAB \tag{A.2}$$

This illustrates that the proportionality constant k in eq (A.2) dictates the speed with which A and B react to produce C, thus this parameter also represents the *reaction rate* or *rate constant* for the given reaction.

In more applicable biochemical situations, many reactions also thermodynamically afford the potential for the reaction to proceed in the reverse direction, which would then be shown as :

$$A + B \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} C \tag{A.3}$$

This equation illustrates that there is a rate constant for both reaction directions. When considering the rate of production of C, or the rate of consumption of either A or B, *both* reaction directions must be considered. The rate at which C is produced is now governed by A and B reacting as well as C decomposing back into A and B. Similar arguments follow for the reactants of the reaction. For example, the rate of change of A can be expressed by:

$$\frac{dA}{dt} = k_{-1}C - k_1AB \tag{A.4}$$

Production of A in the reverse reaction will depend on k_{-1} and C, thereby increasing A, whereas the consumption of A in the forward reaction will depend on k_1, A and B, thus decreasing A.

In general, chemical reactions will take the form:

$$aA + bB \xrightarrow{k} cA + dB \tag{A.5}$$

where a molecules of A and b molecules of B react to produce c molecules of A and d molecules of B.

The rate of change of each reacting component can thus be described by:

$$\frac{dA}{dt} = (c - a)kAB \tag{A.6a}$$
$$\frac{dB}{dt} = (d - b)kAB \tag{A.6b}$$

where kAB represents the speed of reaction, and $c - a$ or $d - b$ represent the stoichiometric coefficients.

A.1.2 Enzyme Kinetics

Enzymes are protein catalysts that facilitate the conversion of substrates into products without undergoing any change themselves. Enzymes accelerate reactions by lowering the energy required for reaction (*the activation energy*). They require a separate method of describing their kinetics beyond that of the law of mass action. This incorporates the production of an intermediate step, where the enzyme E reacts with a substrate S, forming an intermediate complex C. This complex C is then immediately broken down into the product P and the enzyme E. This gives us:

$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} C \xrightarrow{k_2} P + E \tag{A.7}$$

By defining $s = [S], c = C, e = [E]$ and $p = [P]$, this gives the following differential equations by the law of mass action:

$$\frac{ds}{dt} = k_{-1}c - k_1se \tag{A.8a}$$
$$\frac{de}{dt} = (k_{-1} + k_2)c - k_1se \tag{A.8b}$$
$$\frac{dc}{dt} = k_1se - (k_2 + k_{-1})c \tag{A.8c}$$

$$\frac{dp}{dt} = k_2 c \tag{A.8d}$$

Michaelis and Menten then assumed that the substrate forms C at a rate equal to the rate at which it is used up, which steadies over time as the system approaches equilibrium. This assumption is known as the *quasi-steady state assumption*, which implies that $\frac{dc}{dt} = 0$. This assumption is essential to the derivation of enzymatic reactions kinetics, and will be essential is deriving the kinetics of mitochondrial ETC reactions. An additional simplification is achieved by noting that since enzyme is conserved in the reaction, there exists a conservation equation relating the initial amount of enzyme to the amount of enzyme either unbound (e) or bound to substrate (c). Letting e_0 represent the total amount of enzyme present in the system, this means that $e_0 = e + c$. Using these facts, Equation (A.8c) can be rearranged to solve for the amount c at any time t:

$$c(t) = \frac{k_1 e_0 s}{k_1 s + k_{-1} + k_2} \tag{A.9}$$

Using this equation for c, as well as the conservation equation for e_0, eq (A.8a) can then be rearranged:

$$\frac{ds}{dt} = -k_2 c = -\frac{k_1 k_2 e_0 s}{k_1 s + k_{-1} + k_2} \tag{A.10}$$

It is evident that the goal of the reaction is to produce P from the substrate S, so the rate of the overall reaction, or the *speed* of the reaction, can be represented by either production of product or consumption of the substrate. Thus:

$$v(t) = \frac{dp}{dt} = -\frac{ds}{dt} \tag{A.11}$$

which allows for a potential representation for the overall reaction rate using eq (A.10):

$$V_0 = \frac{V_{max} s}{s + K_m} \tag{A.12}$$

where the maximum speed of the reaction is $V_{max} = k_2 e_0$ and the *Michaelis-Menten constant* is $K_m = \frac{k_{-1}+k_2}{k_1}$. Eq (A.12) is known as the *Michaelis-Menten equation*. In eq (A.8d), as k_2 is the only rate constant, and in nearly all cases represents a system where $k_2 \ll k_{-1}$, k_2 is the rate-limiting step and therefore dictates the maximum speed with which the reaction can proceed. Since k_2 by itself is typically so small, it can usually be neglected giving that $K_m = \frac{k_{-1}}{k_1}$. This

Michaelis-Menten constant K_m represents the concentration for s at which the speed of the reaction proceeds at half of the maximum. This is evident if $K_m = s$ in eq (A.12), as it then becomes $V_0 = \frac{s}{2s}V_{max} = \frac{1}{2}V_{max}$. Additionally, at the steady state, since conversions between states are relatively unchanging, K_m approximates the equilibrium constant for the system.

A.1.3 Deriving Two-Substrate Kinetics

The activity of an enzyme following two-substrate kinetics is represented as:

$$E + aA \underset{k_{-a}}{\overset{k_a}{\rightleftharpoons}} E_A \tag{A.13a}$$

$$E + bB \underset{k_{-b}}{\overset{k_b}{\rightleftharpoons}} E_B \tag{A.13b}$$

$$E_B + aA \underset{k_{-a2}}{\overset{k_{a2}}{\rightleftharpoons}} E_{AB} \tag{A.13c}$$

$$E_A + bB \underset{k_{-b2}}{\overset{k_{b2}}{\rightleftharpoons}} E_{AB} \tag{A.13d}$$

$$E_{AB} \xrightarrow{k_p} E + P + Q \tag{A.13e}$$

where substrates A and B can bind to enzyme E to form E_A and E_B, respectively, and result in the production of products P and Q once both substrates are bound as to E as E_{AB}.

Assuming the total amount of enzyme catalyzing this reaction is constant, this means that $E_T = E + E_A + E_B + E_{AB}$ where E_T is constant. Applying the quasi-steady state assumption, we can equate the rate at which enzymes bind and unbind to substrates A, B and to both A and B:

$$k_a E A^a = k_{-a} E_A \tag{A.14a}$$

$$k_b E B^b = k_{-b} E_B \tag{A.14b}$$

From this we see that:

$$E_A = \frac{k_a}{k_{-a}} E A^a \tag{A.15a}$$

$$E_B = \frac{k_b}{k_{-b}} E B^b \tag{A.15b}$$

Using these equations we can then substitute this back into the equation for total enzyme yielding:

$$E_T = E + \frac{k_a}{k_{-a}} EA^a + \frac{k_b}{k_{-b}} EB^b + E_{AB} \tag{A.16}$$

Rearranging gives:

$$E_{AB} = \frac{k_a k_b}{k_{-a} k_{-b}} \Big(\frac{E_T - E_{AB}}{1 + \frac{k_a}{k_{-a}} A^a + \frac{k_b}{k_{-b}} B^b} \Big) AB \tag{A.17}$$

By letting $K_a = \frac{k_a}{k_{-a}}$ and $K_b = \frac{k_b}{k_{-b}}$ this equation can be simplified to:

$$E_{AB} = K_a K_b \Big(\frac{E_T - E_{AB}}{1 + K_b A^a + K_b B^b} \Big) AB \tag{A.18}$$

With rearrangement we get:

$$E_{AB} = \frac{E_T AB}{\frac{1}{K_a K_b} + \frac{A^a}{K_b} + \frac{B^b}{K_a} + AB} \tag{A.19}$$

This equation provides a measure for the amount of enzyme bound to both substrates A and B for any given equilibrium constants and substrate amounts. Thus, in order to find the speed of reaction for producing either product P or Q, we can use the fact that:

$$\frac{dP}{dt} = k_p E_{AB} \tag{A.20}$$

which, in conjunction with equation (A.19) and the fact that $V_{max} = k_p E_T$, the general speed function for the two-substrate model is:

$$v = \frac{V_{max} AB}{\frac{1}{K_a K_b} + \frac{A^a}{K_b} + \frac{B^b}{K_a} + AB} \tag{A.21}$$

A.1.4 Deriving Activation Kinetics

The activity of an enzyme following activation kinetics is represented as:

$$E + aA \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} E_A \tag{A.22a}$$

$$E_A + bS \underset{k_{-2}}{\overset{k_2}{\rightleftharpoons}} E_{AS} \xrightarrow{k_3} E + pP \tag{A.22b}$$

where some activator A can bind to enzyme E to form E_A, only then allowing substrate S to bind to E_A to form product P.

Assuming the total amount of enzyme catalyzing a reaction is constant, $E_T = E + E_A + E_{AS}$, where E_T is constant. Applying the quasi-steady state assumption, it is known that the amount of each enzyme form is unchanging, thus $\frac{dE}{dt} = \frac{dE_A}{dt} = \frac{dE_{AS}}{dt} = 0$. From the law of mass action the following set of equations can be acquired:

$$\frac{dE}{dt} = -k_1 E A^a + k_{-1} E_A + k_3 E_{AS} \tag{A.23a}$$

$$\frac{dE_A}{dt} = k_1 E A^a - k_{-1} E_A - k_2 E_A S^b + k_{-2} E_{AS} \tag{A.23b}$$

$$\frac{dE_{AS}}{dt} = k_2 E_A S^b - k_{-2} E_{AS} - k_3 E_{AS} \tag{A.23c}$$

which are each equivalent to 0 by the quasi-steady state assumption. Rearranging equation (A.23c) gives

$$E_{AS} = \frac{k_2 E_A S^b}{k_{-2} + k_3} \tag{A.24}$$

where $\frac{k_2}{k_{-2} + k_3} = \frac{1}{K_M}$, thus yielding

$$E_{AS} = \frac{E_A S^b}{K_M} \tag{A.25}$$

Similarly for equation (A.23a)

$$E = \frac{k_{-1} E_A + k_3 E_{AS}}{k_1 A^a} \tag{A.26}$$

Applying these back into $E_T = E + E_A + E_{AS}$:

$$E_T = \frac{k_{-1} E_A + k_3 E_{AS}}{k_1 A^a} + E_A + \frac{E_A S^b}{K_M} \tag{A.27}$$

this can be rearranged to get

$$k_1 A^a E_T = \frac{E_{AS} K_M}{S^b}\left(k_{-1} + \frac{S^b}{K_M}(1 + k_3) + 1\right) \tag{A.28}$$

which can then be rearranged to acquire the expression:

$$E_{AS} = \frac{k_3 E_T S^b}{K_M\left(1 + \frac{k_{-1}}{k_1 A^a}\right) + S^b} \tag{A.29}$$

Since k_3 represents the rate for final conversion of $E_A S$ to product, and E_T represents the total amount of enzyme available to catalyze the reaction, $k_3 E_T = V_{max}$. Using this and by letting $K_1 = \frac{k_{-1}}{k_1}$ the rate equation becomes:

$$v = \frac{V_{max} S^b}{K_M\left(1 + \frac{K_1}{A^a}\right) + S^b} \tag{A.30}$$

Appendix B

MATLAB GUI - User Instructions

The readme information shown below is also packaged with the model's code on $http://github.com/Synapt1x$. Access to the private repository housing the model's code can normally be requested by contacting *umcadoni@myumanitoba.ca* or *chriscadonic@gmail.com*, although for the purposes of this masters thesis the code has been supplied.

B.1 GUI Readme

B.1.1 Files

Some of the important files for running this model are: *main.m* and *setup.m*. Main.m starts the program by first generating all of the parameters and relevant model information by calling setup.m. Next the program is displayed by passing in the *parameters* structure generated by setup.m to *finalgui.m*, which controls the display of the GUI and the functionality of all GUI components.

Editing system parameters can be carried out directly in the GUI. For more information on editing within the GUI, see section B.1.3. Parameter values, initial values, and data sources are all directly modifiable in the GUI. For changing default values in any aspect of the model,

however, they must be edited at the time of creation of the *parameters* structure, found in the setup.m function.

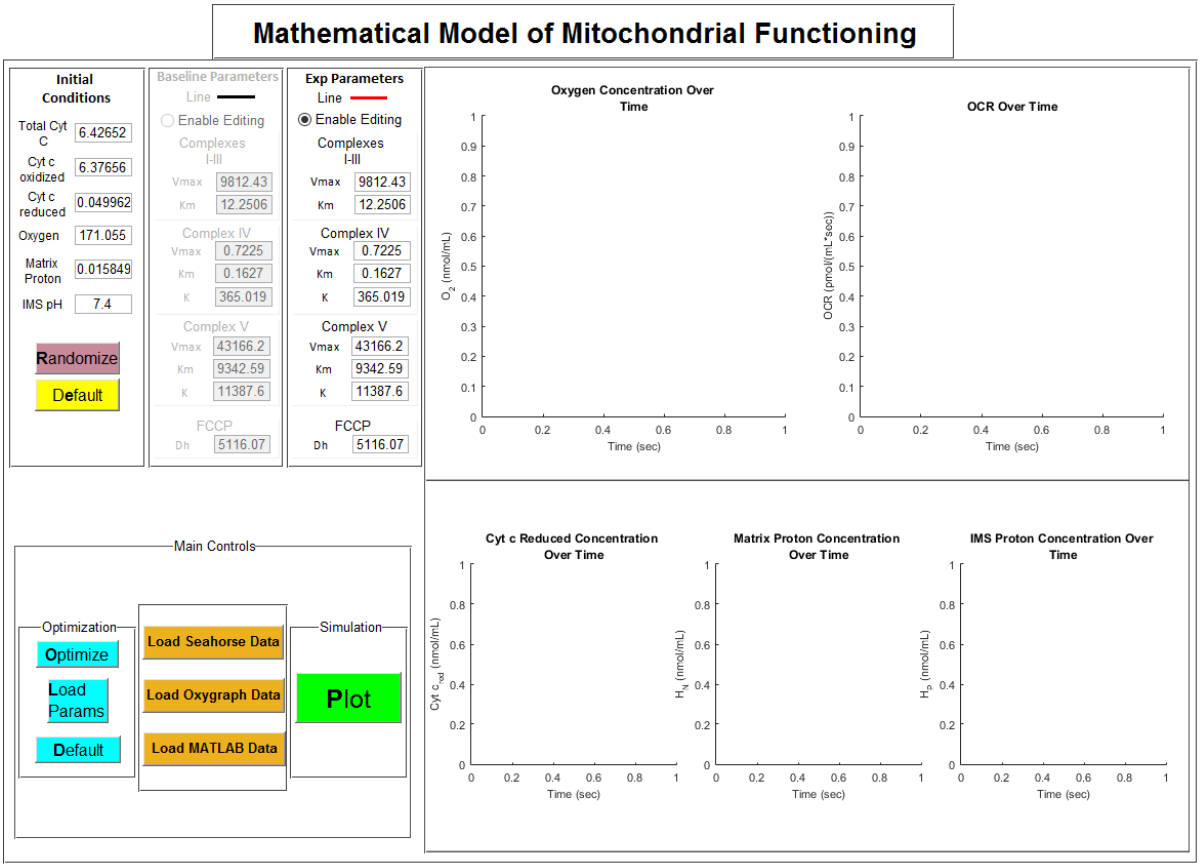
B.1.2 Running the Model

As detailed above, the model can be run simply by navigating to the location of *main.m* and then running this script in MATLAB. The model will be initialized and the GUI will be promptly displayed. Before any simulation is run, the parameters and initial conditions are set to default values. To change these before simulation, they can be directly edited in the text boxes that accompany the value desired. To run a simulation of mitochondrial bioenergetics function, hit the *Plot* button in the bottom section of the GUI. This will solve the differential equations of the model using *ode45* in MATLAB, and then plot the resulting output in the graphs on the right side of the GUI.

B.1.3 The Components of The GUI

In Fig. B.1, the layout of the GUI is shown, including output axes and all input areas. Different aspects of the model can be controlled by the GUI, such as *data sources*, *properties of the model*, and *optimization* of parameters to additional data.

Figure B.1: The graphical user interface of the simulation model as shown when first opened. This shows the interface of the model when the program is first opened. Initial conditions and the parameters of the experimental simulation model are available for editing once the program is run.



- Data source control
  - Located at the bottom section of the input area, there are three buttons for importing and graphing additional data, from either Seahorse XF data, Oroboros Oxygraph data, or from a matlab .mat file
- Model simulation
  - Initial Conditions
    - The user can vary the concentration of each of the listed initial concentrations:
      - Total cytochrome c
      - Cytochrome c oxidized

            - Cytochrome c reduced
            - Oxygen
            - Matrix protons
            - Intermembrane space protons
        * There is a *randomize* button for generating a random set of initial conditions
            - Pressing the **R** key will trigger the randomize button
        * There is a *default* button for resetting the initial conditions to default values
            - Pressing the **E** key will trigger the reset initial conditions button
    - Parameter values for a control condition of the model
        * Parameter values calibrated to experimental data, representing the baseline model without modification
    - Parameter values for an experimental condition of the model
        * The user can edit each parameter value directly, which will update the model values
        * There is also a *default* reset button found in the bottom-left section labeled *Optimization*
            - Pressing the **D** key will trigger the reset parameters button
    - Simulation
        * The user can click the large green *Plot* button to instruct the program to simulate the model, and then display the output of the model in the axes found to the right of the control panels.
            - Pressing the **P** key will trigger the plot function

- Optimization
    - The *optimize* button is used to run *launchQubist.m*, which will run Jason Fiege's *Ferret* genetic algorithm (discussed in Section 2.2.7 and Appendix D) for the purpose of optimizing the parameters to fit additional data.
        * Pressing the **O** key will trigger the optimization button

- The *load params* button allows the user to search for a *-BestResults.mat* file, which is automatically generated by *Ferret* using the post-processing script *analyzeResults.m*. These files are housed within the /DeterministicModel/Solutions/ directory.
  - Pressing the **L** key will trigger the load parameters function
- There is a *default* button found here, as detailed above, for resetting the parameter values in the experimental condition
  - Pressing the **D** key will trigger the reset parameters button

The axes illustrate the behavior of the system once *Plot* is clicked. As shown in Fig. 3.14 in Section 3.3, *Plot* graphs and labels each axis to display the output of the model under the conditions set in the user controls.

B.1.4 Additional Functions of the GUI

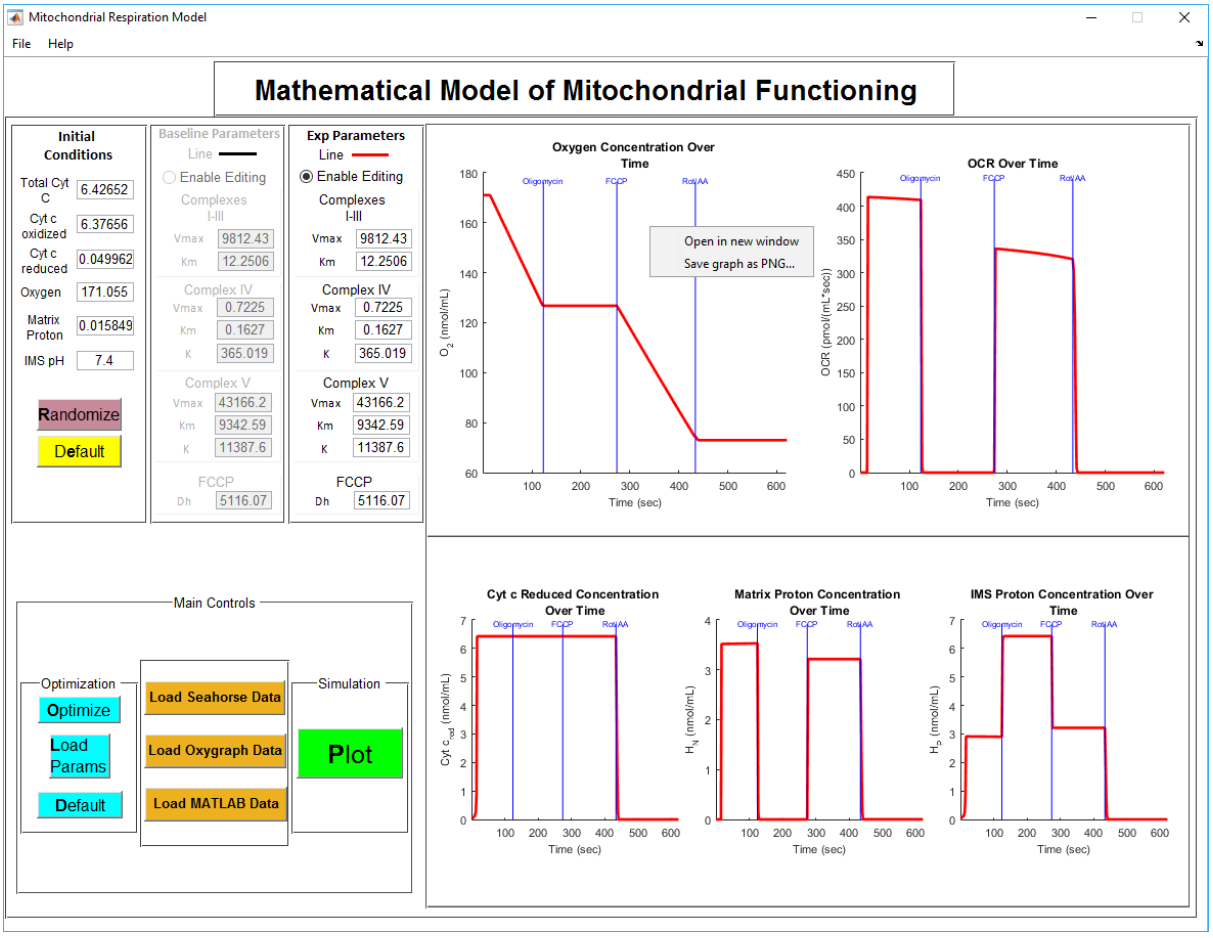
Several additional functions were introduced into the GUI of the program to provide ease-of-use and extend the user's control of the GUI. Each of these functions are discussed in detail in this section.

B.1.5 Right-Clicking a Graph

Each graph, once plotted, can then be right-clicked for additional options as shown in Fig. B.2. These options allow for the user to either save a blown-up image of the plot to a PNG file, or for opening the image in a new figure window for inspection and editing purposes. Both options facilitate creating large images of the graphs for presentation or publication.

Opening the graph in another figure window enables editing and formatting using the built-in MATLAB figure editing functionality, as shown in Fig. B.3. Labeling can be controlled using the figure menus built-in to MATLAB, and the figure can be exported into any format after editing is finalized. Zooming in and out of the graph can also be done in this figure window. The *save graph as PNG...* option simply enlarges the graph to a fullscreen window and saves the graph with enlarged axis labels in a location selected by the user.

Figure B.2: Depiction of the menu available to the user when a graph is right-clicked. The menu will pop-up and allow users to choose to either open the figure in a new, maximized figure window, or save the graph directly to an image file.

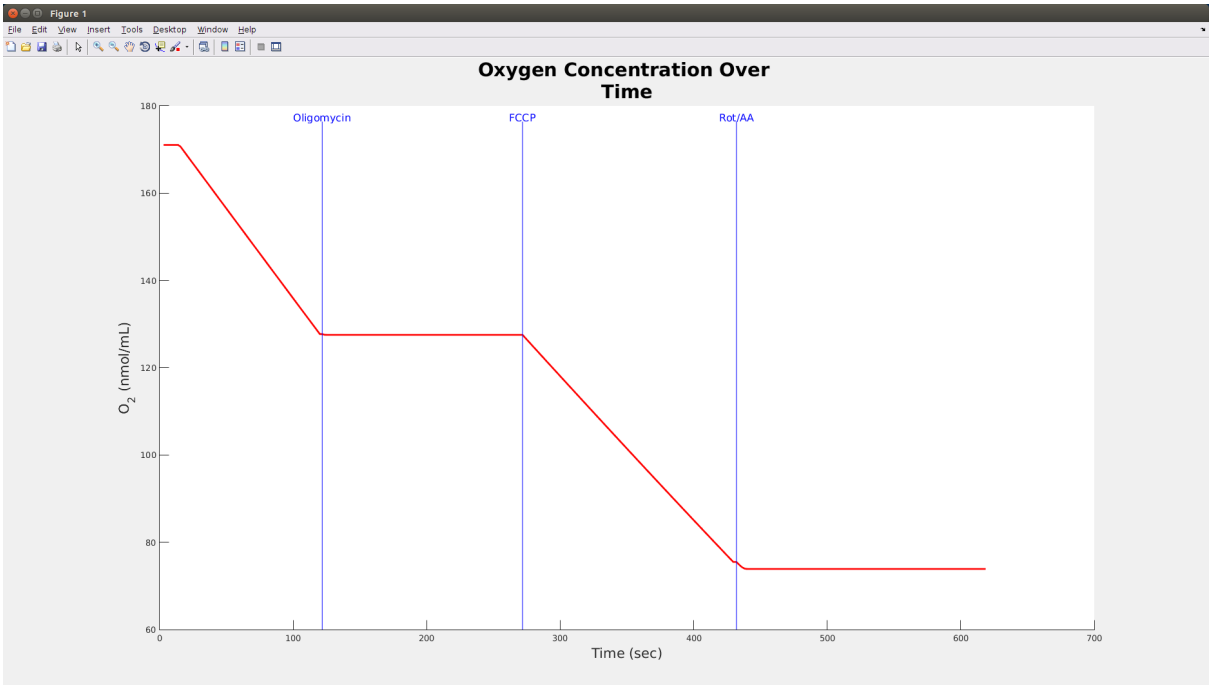


B.1.6 Save Snapshot

In the *File* menu, there is an option to save a screenshot of the entire GUI. The purpose of this is to save the current display of the GUI either for presentation or publication, where the graphs of each element of the system are shown along with the set of parameters and initial conditions that produced those results. Default location for saved images are in the *StateImages* directory, although the user can choose to save the screenshot to a different location.

The keyboard shortcut for this function is *ctrl+d* on windows and linux systems (with windows keyboard layout, not EMACS) and *cmd+d* on MAC OS X systems.

Figure B.3: Opening a graph to be edited in a fullscreen figure window. This allows users to edit and format the appearance of the figure for publication or image generation.



B.1.7 Save Graphs

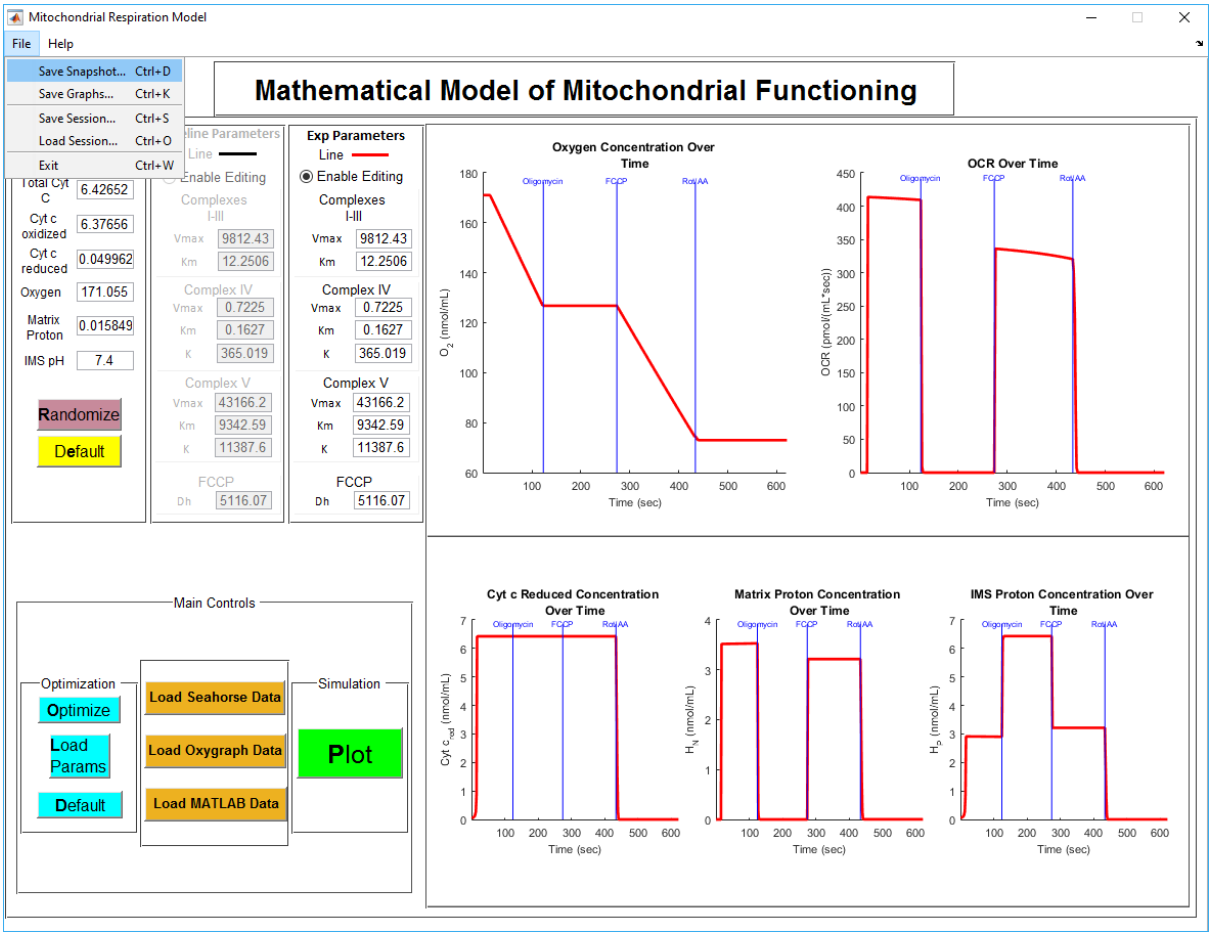
In the *File* menu, there is an option to save a screenshot of just the output of the model. The *Save Graphs...* option allows the user to take a screenshot of just the graphs shown on the right side, and save the .PNG image to a folder of their choice.

The keyboard shortcut for this function is *ctrl+k* on windows and linux systems (with windows keyboard layout, not EMACS) and *cmd+k* on MAC OS X systems.

B.1.8 Save/Load Session

In addition to being able to save a snapshot of the GUI as an image to be viewed or presented later, the state of the model can also be saved. The save and load session commands are found in the *File* menu, with shortcuts *ctrl+s* for saving a session and *ctrl+o* for opening. Fig. B.6 shows these menu options.

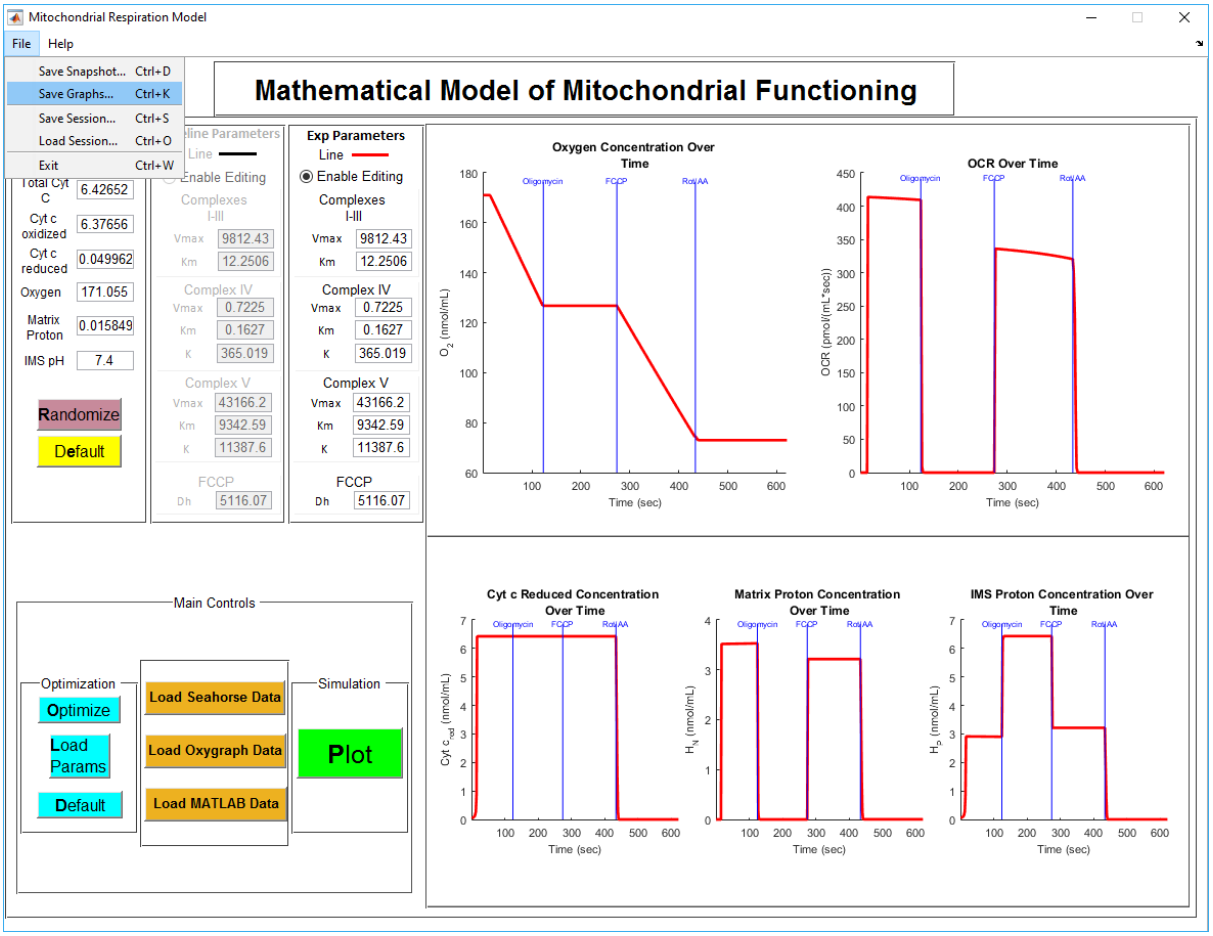
Figure B.4: Depiction of the menu item used to save a snapshot of the entire GUI window to an image. This allows users to store an image of the output of the model alongside the set of model conditions that produced that output. The hotkey for this function is *ctrl+d*.



Saving a session allows the user to save the current values of all variables and all graphics objects in the guidata and handles structures, which will default to save as a .mat file. Once this .mat is saved, it can later be reloaded using the load command giving the user the chance to re-open the state of the chosen file. Saved sessions are generally stored in the *Savestates* directory.

N.B.: Make sure to only load session .mat files when loading a session.

Figure B.5: Depiction of the menu item used to save a snapshot of the model's output to an image. This option allows users to save an image of just the output of the model. The hotkey for this function is *ctrl+k*.



B.1.9 Help Commands

In the *Help* menu, there is a *Version* button and an *Info...* button, as shown in Fig. B.7. The Version button allows the user to confer with the Git system to check the current tagged version of the program. This will inform the user whether or not there are any available updates to the model or the GUI code so that the most up-to-date version of the simulation model can be accessible to the user. The *Info...* button will pull up this README.md document in the appropriate default program.

Figure B.6: Depiction of the menu item used to save and/or load the current properties of the model. This function allows users to save the altered properties, and any additionally saved data, to a MATLAB *.mat* file to continue adjusting simulation properties at a later time by loading the saved *.mat* file. The hotkey for saving the simulation session is ctrl+s, while the hotkey for loading the simulation session is *ctrl+o*.

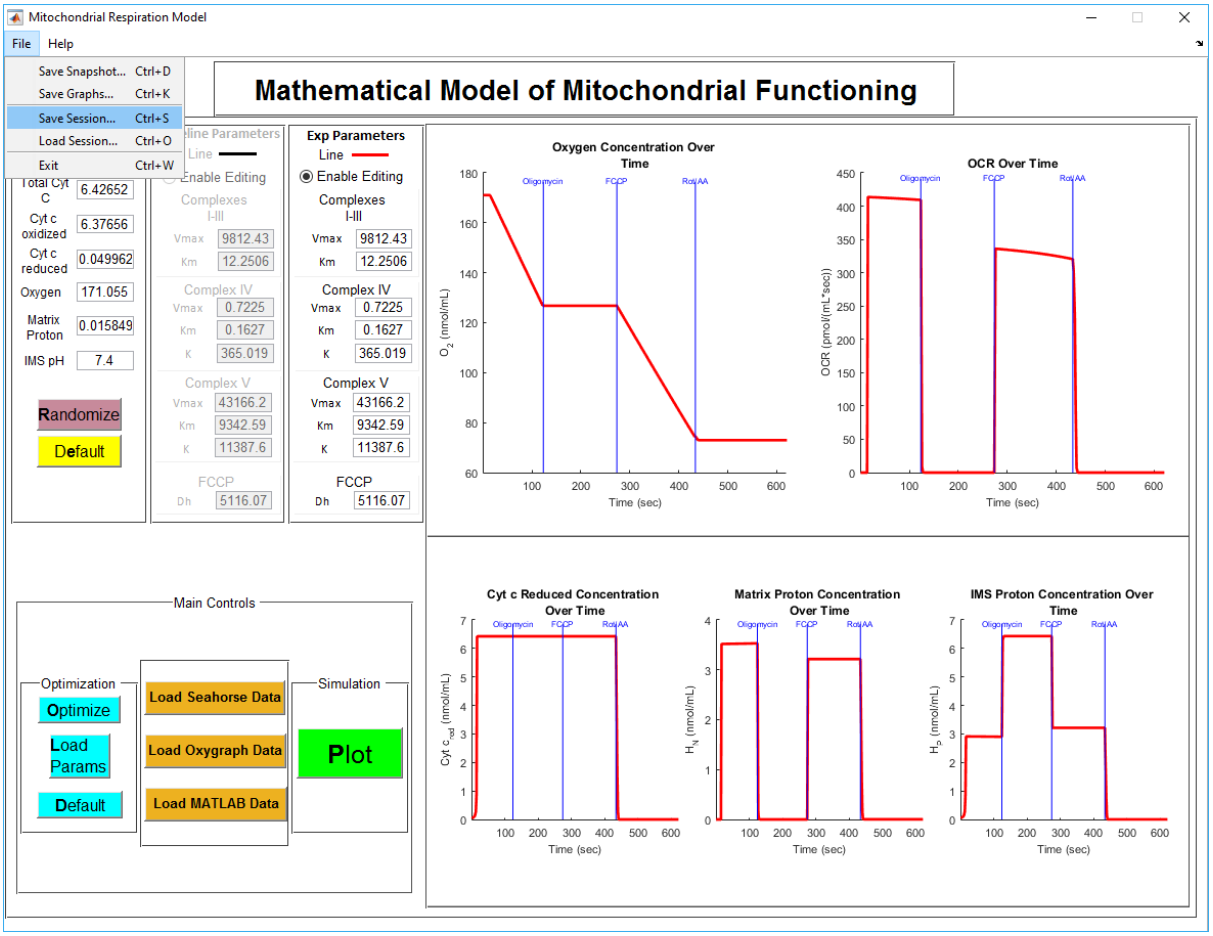
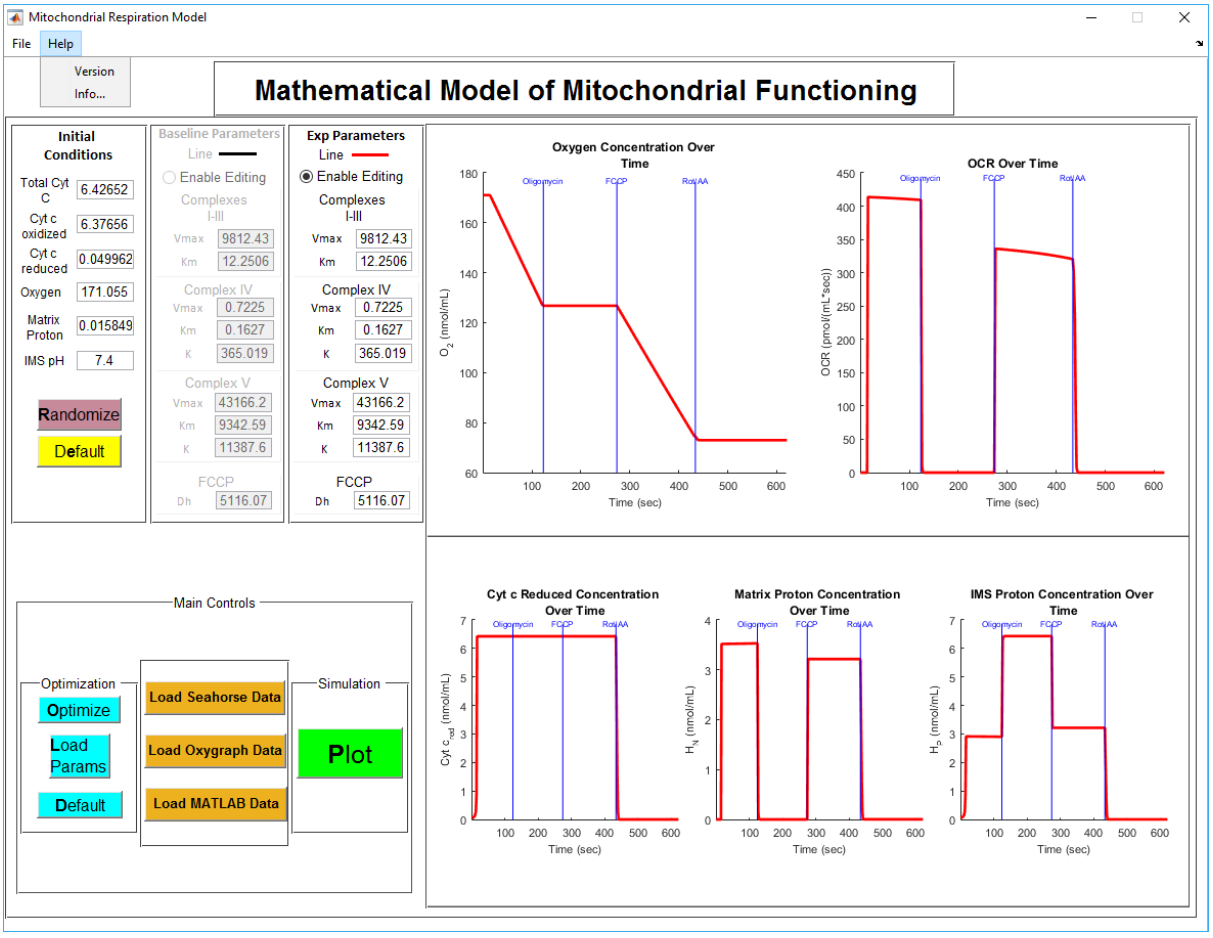


Figure B.7: Depiction of the help menu. This help menu allows users to determine the version number of the current model, using the *git* system to identify if there are any newer version available. The *info...* menu option opens the *readme.md* file in an appropriate program for assistance in using the model.



Appendix C

MATLAB code

C.1 main.m

```
function main
%{
=======================================
Created by: Chris Cadonic
For: M.Sc program in Biomedical Engineering
Project: Modeling Mitochondrial Bioenergetics
=======================================
This is the primary file for running the mitochondrial model created
for my master's project as part of the biomedical engineering
program.

See the readme file 'manual.pdf' for more information as to how
this program functions and how each component script functions.
The readme file 'changelog.txt' indicates the changes implemented
into the model as well as the timestamps for each change.

This code is slightlty altered from its original form to simply model
```

```
the decoupled system, that is, to primarily model complex IV.

For altering the parameters of the model, changes can be made to
the differential equations in 'decoupled_derivative_system.m', the
setup conditions in 'setup.m', optimization run by using
'launchQubist.m', the gui handled by 'main_gui.m', and the
importing of data handled by 'data_formatter.m'.
%}

parameters = setup; %run the setup function which creates the
%structure storing all variables necessary
%for the model (found in 'setup.m')

save parameters %save the model parameters in parameters.mat

%create the GUI for interfacing and display
finalgui(parameters);
```

C.2 setup.m

```
function parameters = setup
%{
Created by: Chris Cadonic
=======================================
The setup function handles the values for each variable in the
system in a structure known as 'parameters'. parameters contains
all of the model's parameters and also the data, graph labels.
%}

%% Data Import
%import the real data
```

```
[parameters.timePoints,parameters.realo2Data, ...
parameters.realOCR] = data_formatter;
parameters.realOCR = parameters.realOCR * 1000; %correct units to pmol/ml s

%% Define the Parameters of the Model
% control condition parameter values
parameters.ctrlParams.Vmax = 0.7225; %bounds: [0.01 10]
parameters.ctrlParams.K1 = 365.0185; %bounds: [0.1 1E4]
parameters.ctrlParams.Km = 0.1627; %bounds: [0.1 1E4]
parameters.ctrlParams.p1 = 43166.2487041382; %bounds: [1 1E4]
parameters.ctrlParams.p2 = 9342.59161533985; %bounds: [1 1E4]
parameters.ctrlParams.p3 = 11387.5724922773; %bounds: [1E-6 1]
parameters.ctrlParams.f0Vmax = 9812.42645440625; %bounds: [0.01 10]
parameters.ctrlParams.f0Km = 12.2505629561911; %bounds: [0.1 1E4]
parameters.ctrlParams.Dh = 5116.07002586063; %bounds: [1E-6 1]
parameters.ctrlParams.cytcred = 0.0499624001853914; %bounds: [1E-6 1]
parameters.ctrlParams.cytcox = 6.37656163806675; %bounds: [1E-6 1]
parameters.ctrlParams.oxygen = parameters.realo2Data(1); %bounds: [1E-6 1]
parameters.ctrlParams.omega = 0.015849; %bounds: [1E-6 1] pH = 7.8
parameters.ctrlParams.rho = 0.0398107; %bounds: [1E-6 1] pH = 7.4

% experimental condition parameter values
% Initially set to be equivalent to the control parameter set
parameters.expParams.Vmax =parameters.ctrlParams.Vmax; %bounds: [0.01 10]
parameters.expParams.K1 =  parameters.ctrlParams.K1; %bounds: [0.1 1E4]
parameters.expParams.Km = parameters.ctrlParams.Km; %bounds: [0.1 1E4]
parameters.expParams.p1 = parameters.ctrlParams.p1; %bounds: [1 1E4]
parameters.expParams.p2 = parameters.ctrlParams.p2; %bounds: [1 1E4]
parameters.expParams.p3 =parameters.ctrlParams.p3; %bounds: [1E-6 1]
parameters.expParams.f0Vmax = parameters.ctrlParams.f0Vmax; %bounds: [0.01 10]
parameters.expParams.f0Km = parameters.ctrlParams.f0Km; %bounds: [0.1 1E4]
```

```
parameters.expParams.Dh = parameters.ctrlParams.Dh; %bounds: [1E-6 1]
parameters.expParams.cytcred = parameters.ctrlParams.cytcred; %bounds: [1E-6 1]
parameters.expParams.cytcox = parameters.ctrlParams.cytcox; %bounds: [1E-6 1]
parameters.expParams.oxygen = parameters.ctrlParams.oxygen; %bounds: [1E-6 1]
parameters.expParams.omega = parameters.ctrlParams.omega; %bounds: [1E-6 1]
parameters.expParams.rho = parameters.ctrlParams.rho; %bounds: [1E-6 1]


%% Define Initial Conditions
%initial conditions in nmol/mL; conversion: 1 nmol/mL = 1E-6 mol/L
parameters.Cytcox = parameters.ctrlParams.cytcox;
parameters.Cytcred = parameters.ctrlParams.cytcred;
parameters.Cytctot = parameters.Cytcox+parameters.Cytcred;
[parameters.ctrlParams.Cytctot,parameters.expParams.Cytctot] = ...
 deal(parameters.Cytctot);
parameters.O2 = parameters.ctrlParams.oxygen;
parameters.Hn = parameters.ctrlParams.omega;

%assuming a pH of 7.4 we get 3.981E-8 mol/L or:
parameters.Hp = parameters.ctrlParams.rho;

%% Define boundary times for integration
%define the time boundaries between conditions; First instance of segment
%change
parameters.oligoTime = min(find(parameters.timePoints>=121.8));
parameters.fccpTime = min(find(parameters.timePoints>=271.8));
parameters.inhibitTime = min(find(parameters.timePoints>=432));

%define the arrays holding the time points for each section
parameters.baselineTimes = parameters.timePoints( ...
1:parameters.oligoTime-1);
```

```
parameters.oligoTimes = parameters.timePoints( ...
parameters.oligoTime:parameters.fccpTime-1);
parameters.fccpTimes = parameters.timePoints( ...
parameters.fccpTime:parameters.inhibitTime-1);
parameters.inhibitTimes = parameters.timePoints( ...
parameters.inhibitTime:end);

%number of points in each section
parameters.numpoints = [numel(parameters.baselineTimes),numel(...
parameters.oligoTimes),numel(parameters.fccpTimes), ...
numel(parameters.inhibitTimes)];

%% Load Additional Functions
% add the additionalFuncs folder to path if it isn't already there
curdir = fileparts(which(mfilename));
addpath([curdir,'/AdditionalFuncs/']);

%% Define the labels and titles for GUI Graphs
%titles and labels for the output graphs
[parameters.title{1:5}] = deal(['Cyt c Reduced Concentration Over'...
' Time'],'Oxygen Concentration Over Time', ...
'OCR Over Time', ...
'Matrix Proton Concentration Over Time',...
'IMS Proton Concentration Over Time');
[parameters.ylab{1:5}] = deal('Cyt c_{red} (nmol/mL)', ...
'O_2 (nmol/mL)','OCR (pmol/(mL*sec))','H_N (nmol/mL)', ...
'H_P (nmol/mL)');
parameters.xlab = 'Time (sec)';
```

C.3 data_formatter.m

```
function [allTimes,realo2,realOCR] = data_formatter
%{
Created by: Chris Cadonic
=======================================
This function reads the excel data files and formats them into
vectors for use in the mitochondria model as calibration data.

This function reads an excel file in a folder called 'Data', found in
the location of this .m file. Data is read and then stored into a data
matrix, with corresponding labels.
%}

%% Read the files for O2 and OCR

%store the folder in which the model is stored
path_folder = fileparts(which(mfilename));

%file names holding the oxygraph o2 data and Seahorse ocr data
filename = fullfile(path_folder, '/Data/oxygraphData.xlsx');

%% Extract Oxygraph Data

%extract all times and all oxygen concentration readings
allData = xlsread(filename,'Sheet1','M520:O829');
allData(1,:)=[]; %delete t=0 time point

%store the times, o2 and ocr data separately
[allTimes,realo2] = deal(allData(:,1),allData(:,2));
```

```
realOCR = -gradient(realo2);
```

C.4 finalgui.m

```
function varargout = finalgui(varargin)
%{
Created by: Chris Cadonic
=====================================
This function handles the user interface for the model, which
provides a GUI that allows the model to be fully accessible. The
user can alter parameters and manipulate the model, run additional
simulations, and thus view how alterations affect the model
without having to restart the model.

The instantiation code and object creation code are generated by
GUIDE in matlab. The remaining code was written specifically to
allow functionality in the program and is specific to the model.
GUIDE was primarily used for the interface, and to lay out elements
All Callback functions were programmed manually for use in this
model.
%}

% Last Modified by GUIDE v2.5 09-Feb-2015 12:53:55

%% Initialization Code
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @main_gui_OpeningFcn, ...
'gui_OutputFcn',  @main_gui_OutputFcn, ...
```

```
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



%% GUI Creation code
% --- Executes just before main_gui is made visible.
function main_gui_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for main_gui
handles.output = hObject;

% get the location of the current directory
handles.curdir = fileparts(which(mfilename));

%take in the setup parameters from the 'main.m' function
if ~isempty(varargin)
handles.parameters = varargin{1};
end

handles.ctrlParams = varargin{1}.ctrlParams;
handles.expParams = varargin{1}.expParams;
```

```
%store the default data for the model
handles.initialData = [handles.parameters.Cytctot, ...
handles.parameters.Cytcox, handles.parameters.Cytcred, ...
handles.parameters.O2, handles.parameters.Hn, ...
handles.parameters.Hp, handles.ctrlParams.Vmax, handles.ctrlParams.K1, ...
handles.ctrlParams.Km, handles.ctrlParams.p1, handles.ctrlParams.p2, ...
handles.ctrlParams.p3, handles.ctrlParams.f0Vmax, handles.ctrlParams.f0Km, ...
handles.ctrlParams.Dh];

%store all graph handles in the handles structure as an array
[handles.graphs{1:5}] = deal(handles.Cytc_plot, ...
handles.O2_plot,handles.OCR_plot,handles.H_N_plot,...
handles.H_P_plot);

%store all control editing text boxes in the handles structure as an array
[handles.allcontEdits{1:9}] = deal(handles.V_max_cedit, handles.K_1_cedit, ...
handles.K_m_cedit,handles.p1_cedit,handles.p2_cedit, handles.p3_cedit, ...
handles.f0Vmax_cedit, handles.f0Km_cedit, handles.Dh_cedit);

%store all exp editing text boxes in the handles structure as an array
[handles.allEdits{1:9}] = deal(handles.V_max_edit, handles.K_1_edit, ...
handles.K_m_edit,handles.p1_edit,handles.p2_edit, handles.p3_edit, ...
handles.f0Vmax_edit, handles.f0Km_edit, handles.Dh_edit);

%store all initial concentrations text boxes in the handles structure as an
%array
[handles.allInitials{1:5}] = deal(handles.initial_cytctot_edit, ...
handles.initial_cytcox_edit, handles.initial_cytcred_edit, ...
handles.initial_o2_edit, handles.initial_hn_edit);
```

```
%label the axes for all graphs
graphLabel(handles);

%insert the initial parameter values into the appropriate textboxes
setParams(handles,handles.initialData(7:end),'control');
setParams(handles,handles.initialData(7:end),'experimental');

%insert the initial concentration values into the textboxes
handles = setInitials(handles, [handles.parameters.Cytctot, ...
handles.parameters.Cytcox, handles.parameters.Cytcred, ...
handles.parameters.O2, handles.parameters.Hn, ...
handles.parameters.Hp]);

%insert the initial conditions into the textboxes
set(findall(handles.controlGroup,'-property','Enable'),'Enable','off');

guidata(hObject,handles);

% --- Outputs from this function are returned to the command line.
function varargout = main_gui_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Main Callback Functions

function finalgui_WindowKeyPressFcn(hObject, eventdata, handles)
% Keypressfcn for the entire GUI
switch eventdata.Key
case {'p','return'}
plot_Callback(hObject,eventdata,handles);
```

```
case 'r'
randomizeButton_Callback(hObject,eventdata,handles);
case 'd'
params_default_Callback(hObject,eventdata,handles);
case 'e'
initial_default_Callback(hObject,eventdata,handles);
case 'o'
optimize_Callback(hObject, eventdata, handles);
case 'l'
loadparams_Callback(hObject, eventdata, handles);
end

function optimize_Callback(hObject, eventdata, handles) %optimize button

%run Qubist for optimization
launchQubist

function initial_cytctot_edit_Callback(hObject,eventdata,handles)
editBox(hObject,handles,'initial','Cytctot');

%get current total Cyt C
currTot = str2double(get(hObject,'String'));
newCytcred = 0;

while ~(newCytcred)
takeVal = inputdlg(['What will be the initial value of Cyt C ', ...
'reduced? The remaining value from the total amount of ', ...
'Cytochrome C will be set as Cyt C oxidized. The New value ', ...
'of Cytochrome C Total: ',num2str(currTot),'.'], ...
'Set Cytochrome Cyt C reduced');
newCytcred = ensureRightInput(str2double(takeVal{1}),currTot);
```

```
end
newCytcox = currTot - newCytcred;

%update the values in boxes and parameters structure
set(handles.initial_cytcox_edit,'String',num2str(newCytcox));
handles.parameters.Cytcox = newCytcox;
set(handles.initial_cytcred_edit,'String',num2str(newCytcred));
handles.parameters.Cytcred = newCytcred;
guidata(hObject,handles);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Edit boxes for Initial conditions and Parameters

function initial_cytcox_edit_Callback(hObject,eventdata,handles)
handles = editBox(hObject,handles,'initial','Cytcox');
guidata(hObject,handles);

function initial_cytcred_edit_Callback(hObject,eventdata,handles)
handles = editBox(hObject,handles,'initial','Cytcred');
guidata(hObject,handles);

function initial_o2_edit_Callback(hObject,eventdata,handles)
handles = editBox(hObject,handles,'initial','O2');
guidata(hObject,handles);

function initial_hn_edit_Callback(hObject,eventdata,handles)
handles = editBox(hObject,handles,'initial','Hn');
guidata(hObject,handles);

function initial_ph_edit_Callback(hObject,eventdata,handles)
getHpconc = 0;
```

```
oldHp = 0;
newHp = 0;
getVal = str2double(get(hObject,'String'));

if isnan(getVal) %if not, throw error box and reset value
msgbox('Please input a valid number.','Not a number');

%get the concentration value for resetting the edit box
getHpconc = getfield(handles.parameters,'Hp');

oldHp = -log10(getHpconc *1E-6);

set(hObject,'String',oldHp);
else %if so, then update the model with new value
%Hp from the given pH
if checkpH(getVal)
newHp = (10^-getVal) * 1E9;
handles.parameters = setfield(handles.parameters,'Hp',newHp);
else
%get the concentration value for resetting the edit box
getHpconc = getfield(handles.parameters,'Hp');

oldHp = -log10(getHpconc *1E-9);
set(hObject,'String',oldHp);
end
end

guidata(hObject,handles);

function V_max_cedit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'control','Vmax');
```

```
guidata(hObject,handles);

function K_1_cedit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'control','K1');
guidata(hObject,handles);

function K_m_cedit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'control','Km');
guidata(hObject,handles);

function p1_cedit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'control','p1');
guidata(hObject,handles);

function p2_cedit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'control','p2');
guidata(hObject,handles);

function p3_cedit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'control','p3');
guidata(hObject,handles);

function f0Vmax_cedit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'control','f0Vmax');
guidata(hObject,handles);

function f0Km_cedit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'control','f0Km');
guidata(hObject,handles);

function Dh_cedit_Callback(hObject, eventdata, handles)
```

```
handles = editBox(hObject,handles,'control','Dh');
guidata(hObject,handles);

function V_max_edit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'experimental','Vmax');
guidata(hObject,handles);

function K_1_edit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'experimental','K1');
guidata(hObject,handles);

function K_m_edit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'experimental','Km');
guidata(hObject,handles);

function p1_edit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'experimental','p1');
guidata(hObject,handles);

function p2_edit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'experimental','p2');
guidata(hObject,handles);

function p3_edit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'experimental','p3');
guidata(hObject,handles);

function f0Vmax_edit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'experimental','f0Vmax');
guidata(hObject,handles);
```

```
function f0Km_edit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'experimental','f0Km');
guidata(hObject,handles);

function Dh_edit_Callback(hObject, eventdata, handles)
handles = editBox(hObject,handles,'experimental','Dh');
guidata(hObject,handles);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Additional buttons
%function for allowing editing in the control parameters
function enableCont_Callback(hObject, eventdata, handles)
if (get(hObject,'Value')==get(hObject,'Max'))
set(findall(handles.controlGroup,'-property','Enable'),'Enable','on');
else
set(findall(handles.controlGroup,'-property','Enable'),'Enable','off');
end;

%function for allowing editing in the control parameters
function enableExp_Callback(hObject, eventdata, handles)
if (get(hObject,'Value')==get(hObject,'Max'))
set(findall(handles.experimentalGroup,'-property','Enable'),'Enable','on');
else
set(findall(handles.experimentalGroup,'-property','Enable'),'Enable','off');
end;

%function for randomizing initial conditions
function randomizeButton_Callback(hObject,eventdata,handles)
%generate random vector
randomVect = randn(1,5)*25+100; % 4 initial conditions
randomVect(3) = randn*0.0033+0.01; % set cyt c red very very low initially
```

```
randomVect(1) = randomVect(2) + randomVect(3); % set cyt c tot to ox + red
randomVect(6) = (10^-(randn*1+7))*1E6; % randomize a pH

%send these values to set Initials to change boxes and parameters
handles = setInitials(handles, randomVect, 'randomize');
guidata(hObject,handles);

%function for resetting initial concentrations
function initial_default_Callback(hObject,eventdata,handles)
handles = setInitials(handles, handles.initialData(1:6), 'setDefault');
guidata(hObject,handles);

%function for resetting initial parameters
function params_default_Callback(hObject, eventdata, handles)
handles = setParams(handles, handles.initialData(7:end), ...
'control','setDefault');
handles = setParams(handles, handles.initialData(7:end),  ...
'experimental','setDefault');
guidata(hObject,handles);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Menu Callback functions
%save the current version of fig
function save_fig_Callback(hObject, eventdata, handles)
%save the image and color map for the overall window
image = getframe(gcf);

try
%save the image to a file specified by the user
[filename,filepath]=uiputfile(fullfile(handles.curdir,'StateImages', ...
[date,'-sessionImage.png']),'Save screenshot file');
```

```
imwrite(image.cdata,[filepath,filename]);

disp(['Image was successfully saved to: ', filepath,filename]);
catch % if an error is caught, don't throw error and instead abort save image
disp('Snapshot save operation aborted.');
end

%save just the figures
function save_graphs_Callback(hObject, eventdata, handles)
% save the image and color map for the overall window
image = getframe(gcf);

% crop just the graphs and store that as the image
image = imcrop(image.cdata,[565,79,817,685]);

try
%save the image to a file specified by the user
[filename,filepath]=uiputfile(fullfile(handles.curdir,'StateImages', ...
[date,'-sessionImage.png']),'Save image of graphs to file');
imwrite(image,[filepath,filename]);

disp(['Image was successfully saved to: ', filepath,filename]);
catch % if an error is caught, don't throw error and instead abort save image
disp('Saving image of graphs aborted.');
end

%save the workspace
function save_session_Callback(hObject,eventdata,handles)
%turn off 'use uisave' warning since uisave is in fact being used
warning('off','MATLAB:Figure:FigureSavedToMATFile');
```

```
try
% save the current data found in the model
currentdata = getappdata(gcf);
[filename,filepath]=uiputfile(fullfile(handles.curdir,'Savestates', ...
[date,'-SaveSession.mat']), 'Save session file');
uisave('currentdata',[filepath,filename]);

disp(['Session was successfully saved session file to: ', filepath,filename]);
catch % if an error is caught, don't throw error and instead abort save session
disp('Session save operation aborted.');
end

%load a saved workspace
function load_session_Callback(hObject,eventdata,handles)
try
[filename,filepath]=uigetfile(fullfile(handles.curdir,'Savestates','*.mat'), ...
'Load session file');
close(gcf);
load([filepath,filename]);

% check if it was a valid session file
if (exist('currentdata'))
disp('Session successfully loaded.');
else % if not, reload finalgui to reset the GUI
disp('.mat chosen was not a correct session savestate. Resetting GUI now.');
finalgui(handles.parameters);
end
catch % if an error is caught, reload finalgui to reset the GUI
disp('Session load operation aborted. Resetting GUI now.');
finalgui(handles.parameters);
end
```

```
function exit_prog_Callback(hObject, eventdata, handles)
disp('Goodbye! Thank you for using my mitochondrial model!');
close;

function version_Callback(hObject, eventdata, handles)
try
[~,ver]=system('git describe --abbrev=0');
msgbox(['The current version of this code is ',ver(1:end-1),'.'], ...
'Code Version');
catch
mshbox('To check the code version, "git" is required.','Git not found');
end

function info_Callback(hObject,eventdata, handles)
cd ..; % go up one directory level
open('README.md'); %open the readme file
cd DeterministicModel; % Re-enter DeterministicModel directory

function save_graph_Callback(hObject, eventdata, handles)
%output the figure to be saved
newgraph = openGraph('save');

%acquire the desired name for the figure
[figname,figpath]=uiputfile('.png','Please save the figure file.');

%save figure into fig file pointed out by the user
if ischar(figname) %check if user selected an output name
set(newgraph,'color','w');
export_fig(newgraph,[figpath,figname]);
else %if not, then abort saving and provide message
```

```
msgbox('No output file name provided.','Operation aborted.');
end
%close the figure to free memory
close(newgraph);

function open_graph_Callback(hObject, eventdata, handles)
openGraph; %simply open the figure in a new window

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plot Graphs Callback
function plot_Callback(hObject, eventdata, handles) %plot button in gui

%store variables for differntiating control and experimental parameter sets
graphColor = {'black','r'};
types = {'control','experimental'};
params = {handles.ctrlParams,handles.expParams};

%clear all axes graphs using arrayfun to distribute cla to each axes
arrayfun(@cla,findall(0,'type','axes'))

for type=1:2

%plug in the equations into the ode solver
[t,y] = solver(handles.parameters,params{type});

%store the values calculated for each variable
[cytcred, o2, Hn, Hp] = deal(y(:,1),y(:,2),y(:,3),y(:,4));

%calculate the OCR values from the oxygen
calcOCR = calculateOCR(handles,cytcred,o2,Hn,Hp,types{type});
calcOCR = calcOCR * 1000;
```

```
%plot the Cyt c concentration over time
axes(handles.Cytc_plot);
hold on
plot(t(2:end),cytcred(2:end),graphColor{type},'lineWidth',2);
hold off

%plot the O2 concentration over time with real O2 data on top
axes(handles.O2_plot);
hold on
plot(t(2:end),o2(2:end),graphColor{type},'lineWidth',2);
hold off

%plot the OCR over time with real OCR data on top
axes(handles.OCR_plot);
hold on
plot(t(2:end),calcOCR(2:end),graphColor{type},'lineWidth',2);
hold off

%plot the Hn concentration over time
axes(handles.H_N_plot);
hold on
plot(t(2:end),Hn(2:end),graphColor{type},'lineWidth',2);
hold off

%plot the Hp concentration over time
axes(handles.H_P_plot);
hold on
plot(t(2:end),Hp(2:end),graphColor{type},'lineWidth',2);
hold off
```

```
end

%add vertical lines to all graphs for injection times
for graph = 1:numel(handles.graphs)
axes(handles.graphs{graph});
vertScale = get(gca,'yLim'); % get the y resolution
vertRange = [vertScale(1), vertScale(end)*0.98];

% draw oligo line
line([handles.parameters.oligoTimes(1), handles.parameters.oligoTimes(1)], ...
vertRange, 'Color','b','LineWidth',0.01);
text(handles.parameters.oligoTimes(1),vertRange(end)*1.005,'Oligomycin', ...
'FontSize',6,'HorizontalAlignment','center','Color','b');

% draw fccp line
line([handles.parameters.fccpTimes(1), handles.parameters.fccpTimes(1)], ...
vertRange,'Color','b');
text(handles.parameters.fccpTimes(1),vertRange(end)*1.005,'FCCP', ...
'FontSize',6,'HorizontalAlignment','center','Color','b');

% draw inhibit line
line([handles.parameters.inhibitTimes(1), ...
 handles.parameters.inhibitTimes(1)], vertRange, 'Color','b');
text(handles.parameters.inhibitTimes(1),vertRange(end)*1.005,'Rot/AA', ...
'FontSize',6,'HorizontalAlignment','center','Color','b');

% while iterating over graphs, also set xLim
set(gca,'xLim',[t(1), t(end)]);

end
```

```
%update all the graph axes
graphLabel(handles);

guidata(hObject,handles);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Graph Labeling Function
function graphLabel(handles)
%{
since updating the axes elements resets the axis properties such as title,
this function is called each time a figure is plotted so as to reset the
titles and labels to the proper text.
%}
for i=1:numel(handles.parameters.title)
axes(handles.graphs{i})
set(handles.graphs{i},'FontSize',8);
xlabel(handles.parameters.xlab,'FontName','Helvetica','FontSize',8);
ylabel(handles.parameters.ylab{i},'FontName','Helvetica','FontSize',8);
title(textwrap({handles.parameters.title{i}},30), ...
'FontWeight','bold','FontName','Helvetica','FontSize',9);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Load Previous Solutions Button
function loadparams_Callback(hObject,eventdata,handles)

%open dialog for user to navigate to file
[filename,filepath] = uigetfile(fullfile(handles.curdir, ...
'Solutions','*-BestResults.mat'),['Select the "BestResults.mat"', ...
```

```
'containing the parameter set to load']);

if ischar(filename) %if a file is selected, load that file
load([filepath,filename]); %load the file

%change all the values of parameters to loaded parameter set
handles = setParams(handles,myResults','experimental','changeVals');
%additional argin signals setParams to update handles.parameters
guidata(hObject,handles);
else
disp('No file selected. Load parameters operation aborted.');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Open Clicked Figure in New Figure
function varargout = openGraph(varargin)
%determine which object was clicked
whichgraph = gco;
obj=get(gca);
% set(whichgraph,'DefaultPlotFontSize',16);

%open a new figure using the graph from the relevant axes
h2copy = allchild(whichgraph); %extract all children from hObject
if isempty(h2copy) %check to see if the graph exists yet
msgbox(['This function has not been plotted yet. Please use the ', ...
'plot button below to graph the function before opening it.'],'No Plot');
else
if ~isempty(varargin)
% create the figure
newgraph = figure('Visible','Off','units','normalized','outerposition', ...
[0 0 1 1]);
```

```
else
% create the figure
newgraph = figure('units','normalized','outerposition',[0 0 1 1]);
end
hParent = axes; %create handle for axes child
copyobj(h2copy,hParent) %copy the original graph to the new fig

%now add the correct labels to the new figure
xlabel(obj.XLabel.String,'FontName','Calibri','FontSize',16);
ylabel(obj.YLabel.String,'FontName','Calibri','FontSize',16);
title(obj.Title.String,'FontSize',22,'FontWeight','bold','FontName', ...
'Calibri');

%change the children to change the reagent text sizes
textChildren = findobj(hParent,'FontSize',6); % get the text objects
set(textChildren,'FontSize',12); % increase their font size

%optionally output the figure for the 'save' feature
varargout{1}=newgraph;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Change all parameter values
function handles = setParams(handles,values,type,varargin)
%insert the parameter values passed to the function in the GUI

%check for whether it is control or experimental parameters
if strcmp(type,'control')
boxes = handles.allcontEdits;
params = handles.ctrlParams;
```

```
else
boxes = handles.allEdits;
params = handles.expParams;
end

%loop over and change all the displayed values for the parameters
for i = 1:numel(boxes)
set(boxes{i},'String',values(i));
end

%change all the values in the correct params struc if vargin nonempty
if ~isempty(varargin)
[params.Vmax, params.K1, params.Km, params.p1, params.p2, params.p3, ...
params.f0Vmax, params.f0Km, params.Dh] = deal(values(1), values(2), ...
values(3), values(4), values(5),values(6),values(7),values(8), values(9));
end

if strcmp(type,'control')
handles.ctrlParams = params;
else
handles.expParams = params;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Change all Initial values
function handles = setInitials(handles,values,varargin)
%insert the parameter values passed to the function in the GUI

%loop over and change all the displayed values for the parameters
for i = 1:numel(handles.allInitials)
set(handles.allInitials{i},'String',values(i));
```

```
end

%calc pH from concentration and set the proper text box to it
setpH=-log10(values(6)*1E-6);
set(handles.initial_ph_edit,'String',setpH);

%change all the values in the handles.parameters struc if vargin nonempty
if ~isempty(varargin)
[handles.parameters.Cytctot, handles.parameters.Cytcox, ...
handles.parameters.Cytcred, handles.parameters.O2, ...
handles.parameters.Hn, handles.parameters.Hp] = deal( ...
values(1), values(2), values(3), values(4), values(5), values(6));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Edit text box
function handles = editBox(hObject,handles,type,paramChange)
%extract the new value input by the user
newVal = str2double(get(hObject, 'String'));

if strcmp(type,'control')
%check for whether or not a correct input was given
if isnan(newVal) %if not, throw error box and reset value
msgbox('Please input a valid number.','Not a number');
set(hObject,'String',getfield(handles.ctrlParams,paramChange));
else %if so, then update the model with new value
handles.ctrlParams = setfield(handles.ctrlParams,paramChange,newVal);
end
elseif strcmp(type,'experimental')
%check for whether or not a correct input was given
if isnan(newVal) %if not, throw error box and reset value
```

```
msgbox('Please input a valid number.','Not a number');
set(hObject,'String',getfield(handles.expParams,paramChange));
else %if so, then update the model with new value
handles.expParams = setfield(handles.expParams,paramChange,newVal);
end
else
%check for whether or not a correct input was given
if isnan(newVal) %if not, throw error box and reset value
msgbox('Please input a valid number.','Not a number');
set(hObject,'String',getfield(handles.parameters,paramChange));
else %if so, then update the model with new value
handles.parameters = setfield(handles.parameters,paramChange,newVal);
end
end

%also check to see if cytochrome c total needs to be updated
if strcmp(paramChange,'Cytcred')|strcmp(paramChange,'Cytcox')
%update the total amount of cytochrome c total
handles = updateInitialCytctot(hObject,handles);
guidata(hObject,handles);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Update Initial Cytchrome C Total
function handles = updateInitialCytctot(hObject,handles)
%get current total cyt c
newCytcox = str2double(get(handles.initial_cytcox_edit,'String'));
newCytcred = str2double(get(handles.initial_cytcred_edit,'String'));
newTot = newCytcox + newCytcred;

%increase cyt c tot by the amount of introduced cyt c red
```

```
set(handles.initial_cytctot_edit,'String',newCytcox+newCytcred);
handles.parameters.Cytctot = newTot;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Quick function for calculating OCR from o2
function ocr = calculateOCR(handles,cytcred,o2,Hn,Hp,type)

% check whether this calculation is for control or experimental parameters
if strcmp(type,'control')
params = handles.ctrlParams;
else
params = handles.expParams;
end

ocr = (1/2).*((params.Vmax.*o2)./(params.Km.*(1+(params.K1./cytcred))+o2)) ...
.*Hn./Hp;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Check for input value
function cytcred = ensureRightInput(input,currTot)
if ~isnumeric(input)
msgbox('Not a valid number. Please enter a number.','Not a number');
else
if input > currTot
waitfor(msgbox(['Please enter a number less than the ', ...
'total amount of Cytochrome C. That is, less than ', ...
num2str(currTot),'.'], 'Cytochrome C reduced level too high'));
cytcred = 0;
else
cytcred = input;
end
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Check for valid pH value
function validity = checkpH(value)
validity = true;
if (value < 0) || (value > 14)
waitfor(msgbox('Not a valid pH.','Invalid pH'));
validity = false;
end
```

C.5 baselineSystem.m

```
function dy = baselineSystem(t,y,params)
%{
Created by: Chris Cadonic
=======================================
This function maintains all the baseline derivatives
relevant to my masters project.

%}

%input all our variables into the state variable y
Cytcred = y(1);
O2 = y(2);
Hn = y(3);
Hp = y(4);

%{
To decouple the system, complexes I-III activity is instead
approximated by ((parameters.Vmax.*(cytcdiff))./ ...
```

```
(parameters.Km+(cytcdiff))).*(Hn./Hp)

Given this, conservation occurs between NADH and NAD, Succ and
Fum, Q and QH2. Since ((parameters.Vmax.*(cytcdiff))./ ..
(parameters.Km+(cytcdiff))).*(Hn./Hp) approximates BOTH
forward and reverse we get consumption and production of each
component in these pairs as equivalent. Thus the other substrates
do not change in concentration, and we have their time derivatives
equal to 0.

For the baseline conditions, these are the full equations (without
FCCP terms in dy(3) and dy(4))
Both cytochrome c reduced and omega have been reduced to order
1 due to the constraint that cyt c delivers electrons one at a time

Also, to incorporate all sections of the data, time points will dictate
the set of equations used for the model. From the data file: oligo
is injected at t = 18.6 m, FCCP starts injection at t = 20.17 m, and
rot/AA start injection at t = 28.13 m.
%}
cytcdiff = params.Cytctot - Cytcred;

dy(1) = 2*((params.f0Vmax*(cytcdiff))/(params.f0Km+(cytcdiff))) ...
*(Hn./Hp) - 2*((params.Vmax*O2)/(params.Km*(1 ...
+(params.K1/Cytcred))+O2))*(Hn/Hp); %dCytcred
dy(2) = -0.5*((params.Vmax*O2)/(params.Km* ...
(1+(params.K1/Cytcred))+O2))*(Hn/Hp); %dO2
dy(3) = -6*((params.f0Vmax*(cytcdiff))/(params.f0Km+(cytcdiff))) ...
*(Hn./Hp) - 4*((params.Vmax*O2)/(params.Km*(1 ...
+(params.K1/Cytcred))+O2))*(Hn/Hp) + ((params.p1.*Hp) ...
/(Hp+params.p2.*Hn+params.p3)).*Hp; %dHn
```

```
dy(4) = 8*((params.f0Vmax*(cytcdiff))/(params.f0Km+(cytcdiff))) ...
*(Hn./Hp) + 2*((params.Vmax*O2)/(params.Km*(1 ...
+(params.K1/Cytcred))+O2))*(Hn/Hp) - ((params.p1.*Hp) ...
/(Hp+params.p2.*Hn+params.p3)).*Hp; %dHp

dy=dy'; %correct vector orientation

end
```

C.6 oligoSystem.m

```
function dy = oligoSystem(t,y,params)
%{
Created by: Chris Cadonic
=======================================
This function maintains all the oligomycin derivatives
relevant to my masters project.

%}

%input all our variables into the state variable y
Cytcred = y(1);
O2 = y(2);
Hn = y(3);
Hp = y(4);

%{
To decouple the system, complexes I-III activity is instead
approximated by ((parameters.Vmax.*(cytcdiff))./ ...
(parameters.Km+(cytcdiff))).*(Hn./Hp)
```

```
Given this, conservation occurs between NADH and NAD, Succ and
Fum, Q and QH2. Since ((parameters.Vmax.*(cytcdiff))./ ...
(parameters.Km+(cytcdiff))).*(Hn./Hp) approximates BOTH
forward and reverse we get consumption and production of each
component in these pairs as equivalent. Thus the other substrates
do not change in concentration, and we have their time derivatives
equal to 0.

For the oligomycin conditions, these are the full equations (without
FCCP terms in dy(3) and dy(4)) and without ATP Synthase equations
Both cytochrome c reduced and omega have been reduced to order
1 due to the constraint that cyt c delivers electrons one at a time

Also, to incorporate all sections of the data, time points will dictate
the set of equations used for the model. From the data file: oligo
is injected at t = 18.6 m, FCCP starts injection at t = 20.17 m, and
rot/AA start injection at t = 28.13 m.
%}
cytcdiff = params.Cytctot - Cytcred;

dy(1) = 2*((params.f0Vmax*(cytcdiff))/(params.f0Km+(cytcdiff))) ...
*(Hn./Hp) - 2*((params.Vmax*O2)/(params.Km*(1 ...
+(params.K1/Cytcred))+O2))*(Hn/Hp); %dCytcred
dy(2) = -0.5*((params.Vmax*O2)/(params.Km*(1+(params.K1 ...
/Cytcred))+O2))*(Hn/Hp); %dO2
dy(3) = -6*((params.f0Vmax*(cytcdiff))/(params.f0Km+(cytcdiff))) ...
*(Hn./Hp) - 4*((params.Vmax*O2)/(params.Km*(1 ...
+(params.K1/Cytcred))+O2))*(Hn/Hp);%dHn
dy(4) = 8*((params.f0Vmax*(cytcdiff))/(params.f0Km+(cytcdiff))) ...
*(Hn./Hp) + 2*((params.Vmax*O2)/(params.Km*(1 ...
+(params.K1/Cytcred))+O2))*(Hn/Hp); %dHp
```

```
dy=dy'; %correct vector orientation

end
```

C.7 fccpSystem.m

```
function dy = fccpSystem(t,y,params)
%{
Created by: Chris Cadonic
=========================================
This function maintains all the FCCP derivatives
relevant to my masters project.

%}

%input all our variables into the state variable y
Cytcred = y(1);
O2 = y(2);
Hn = y(3);
Hp = y(4);

%{
To decouple the system, complexes I-III activity is instead
approximated by ((parameters.Vmax.*(cytcdiff))./ ...
(parameters.Km+(cytcdiff))).*(Hn./Hp)

Given this, conservation occurs between NADH and NAD, Succ and
Fum, Q and QH2. Since ((parameters.Vmax.*(cytcdiff))./ ...
(parameters.Km+(cytcdiff))).*(Hn./Hp) approximates BOTH
forward and reverse we get consumption and production of each
```

```
component in these pairs as equivalent. Thus the other substrates
do not change in concentration, and we have their time derivatives
equal to 0.

For the conditions following FCCP injection, these are the full
equations (with FCCP terms in dy(3) and dy(4)).
Both cytochrome c reduced and omega have been reduced to order
1 due to the constraint that cyt c delivers electrons one at a time

Also, to incorporate all sections of the data, time points will dictate
the set of equations used for the model. From the data file: oligo
is injected at t = 18.6 m, FCCP starts injection at t = 20.17 m, and
rot/AA start injection at t = 28.13 m.
%}
cytcdiff = params.Cytctot - Cytcred;

dy(1) = 2*((params.f0Vmax*(cytcdiff))/(params.f0Km+(cytcdiff))) ...
*(Hn./Hp) - 2*((params.Vmax*O2)/(params.Km*(1 ...
+(params.K1/Cytcred))+O2))*(Hn/Hp); %dCytcred
dy(2) = -0.5*((params.Vmax*O2)/(params.Km*(1+(params.K1 ...
/Cytcred))+O2))*(Hn/Hp); %dO2
dy(3) = -6*((params.f0Vmax*(cytcdiff))/(params.f0Km+(cytcdiff))) ...
*(Hn./Hp) - 4*((params.Vmax*O2)/(params.Km*(1 ...
+(params.K1/Cytcred))+O2))*(Hn/Hp)+ params.Dh ...
* ((Hp - Hn) + Hp * log(Hp/Hn)); %dHn
dy(4) = 8*((params.f0Vmax*(cytcdiff))/(params.f0Km+(cytcdiff))) ...
*(Hn./Hp) + 2*((params.Vmax*O2)/(params.Km*(1 ...
+(params.K1/Cytcred))+O2))*(Hn/Hp) - params.Dh ...
* ((Hp - Hn) + Hp * log(Hp/Hn)); %dHp

dy=dy'; %correct vector orientation
```

```
end
```

C.8 inhibitSystem.m

```
function dy = inhibitSystem(t,y,params)
%{
Created by: Chris Cadonic
=======================================
This function maintains all the inhibited system derivatives
relevant to my masters project.

%}

%input all our variables into the state variable y
Cytcred = y(1);
O2 = y(2);
Hn = y(3);
Hp = y(4);

%{
To decouple the system, complexes I-III activity is instead
approximated by ((parameters.Vmax.*(cytcdiff))./ ...
(parameters.Km+(cytcdiff))).*(Hn./Hp)*(Hn/Hp)

Given this, conservation occurs between NADH and NAD, Succ and
Fum, Q and QH2. Since ((parameters.Vmax.*(cytcdiff))./ ...
(parameters.Km+(cytcdiff))).*(Hn./Hp)*(Hn/Hp) approximates BOTH forward
and reverse we get consumption and production of each
component in these pairs as equivalent. Thus the other substrates
do not change in concentration, and we have their time derivatives
```

```
equal to 0.

For the baseline conditions, these are the full equations (without
FCCP terms in dy(3) and dy(4))
Both cytochrome c reduced and omega have been reduced to order
1 due to the constraint that cyt c delivers electrons one at a time

Also, to incorporate all sections of the data, time points will dictate
the set of equations used for the model. From the data file: oligo
is injected at t = 18.6 m, FCCP starts injection at t = 20.17 m, and
rot/AA start injection at t = 28.13 m.
%}

dy(1) = -2*((params.Vmax*O2) ...
/(params.Km*(1+(params.K1/Cytcred))+O2))...
*(Hn/Hp); %dCytcred
dy(2) = -0.5*((params.Vmax*O2) ...
/(params.Km*(1+(params.K1/Cytcred))+O2))...
*(Hn/Hp); %dO2
dy(3) =  -4*((params.Vmax*O2)/(params.Km*(1 ...
+(params.K1/Cytcred))+O2))*(Hn/Hp) + params.Dh ...
* ((Hp - Hn) + Hp * log(Hp/Hn)); %dHn
dy(4) =  2*((params.Vmax*O2)/(params.Km*(1 ...
+(params.K1/Cytcred))+O2))*(Hn/Hp) - params.Dh ...
* ((Hp - Hn) + Hp * log(Hp/Hn)); %dHn

dy=dy'; %correct vector orientation

end
```

C.9 solver.m

```
function [t,y] = solver(parameters,params)
%{
Created by: Chris Cadonic
=======================================
This function solves the full situation for my model by step-wise
solving the ODEs for each section using the appropriate equations.
%}

%update all parameter values
parameters.Cytcred = params.cytcred;
parameters.Cytcox = params.cytcox;
parameters.Cytctot = parameters.Cytcred + parameters.Cytcox;
parameters.Hn = params.omega;
parameters.Hp = params.rho;
parameters.O2 = params.oxygen;
params.Cytctot = params.cytcred + params.cytcox;

%Set the options for running ode45
options = odeset('NonNegative',[1,2,3,4]);

%Solve by using ode for each section and passing along the final
%values as initial values for the next section
tic
[t1,y1] = ode45(@baselineSystem, parameters.baselineTimes, ...
[params.cytcred,params.oxygen,params.omega,params.rho],options,params);
[t2,y2] = ode45(@oligoSystem, parameters.oligoTimes, ...
[y1(end,1),y1(end,2),y1(end,3),y1(end,4)],options,params);
[t3,y3] = ode45(@fccpSystem, parameters.fccpTimes, ...
[y2(end,1),y2(end,2),y2(end,3),y2(end,4)],options,params);
```

```
[t4,y4] = ode45(@inhibitSystem, parameters.inhibitTimes, ...
[y3(end,1),y3(end,2),y3(end,3),y3(end,4)],options,params);
toc

t = [t1;t2;t3;t4];
y = [y1;y2;y3;y4];
```

C.10 fitness.m

```
function F = fitness(X,extPar) %This function evaluates the
%fitness for the input solving agent

parameters=extPar.parameters;
params=extPar.parameters.expParams;
f=fields(X);
f(strcmpi(f,'info'))=[];

for n=length(X):-1:1
if isAbortEval(extPar.status)
F=[];
break
end

for i=1:length(f)
params.(f{i})=X(n).(f{i});
end

%call ode to solve the system of equations for this solver
[t, y] = solver(parameters,params);

%for fitting O2
```

```
evaluations = y(:,2); %evaluated data for o2
realo2Data = parameters.realo2Data; %use actual o2 data

%evaluate using a least-squares
F(1,n) = sum((realo2Data-evaluations).^2)/numel(realo2Data);
pause(0.001);

end
```

C.11 analyzeResults.m

```
function analyzeResults(OptimalSolutions)
%{
Created by: Chris Cadonic
=======================================
As per code generation suggestions in the Qubist manual, this function
can be called by Ferret after analysis to 'automatically post-process'
results.

In this function, OptimalSolutions is loaded and then it will be saved to
the 'Solutions' folder in the format "Date-OptimalSolutions.mat".
This structure will then be analyzed in the code below to look for the
global best, which will be stored in the variable result.
%}

%% Initialize vars

%initialize storage variables
myResults = inf;
bestFit = OptimalSolutions.F(:,1).*inf;
```

```
%% Loop over OptimalSolutions to find Best

% To loop over entire optimal set:
for n=1:size(OptimalSolutions.X, 2) % Number of columns = number of solutions.

%first use bsxfun to check 'greater than' for all elements of bestFit
%vs. OptimalSolutions.F
checkSize = bsxfun(@gt,bestFit,OptimalSolutions.F(:,n));

if all(checkSize) %if current best is greater than optimal then replace bestFit
%all is used to check to see if all F-values are less than best
myResults=OptimalSolutions.X(:,n);
bestFit=OptimalSolutions.F(:,n);
end
end

%% Save files to Solutions folder

folder = fileparts(which(mfilename)); %get the current folder
cd([folder '/Solutions']); %change to Solutions folder
todayDate = date; %get the run date

%save the Best solution to the Solutions folder
resultsname = [todayDate '-BestResults'];
save(resultsname,'myResults','bestFit');

%display a message indicating the files will be saved
disp(['Saving output files to ' folder '/Solutions.']);

cd(folder); %change back to original folder
```

C.12 sensitivityAnalysis.m

```
function sensitivityVals = sensitivityAnalysis()
%{
Created by: Chris Cadonic
=======================================
This function carries out a relative sensitivty analysis for the entire system
using the method outlined in Beard (2005).

E_star stores the error values when the model parameter values
are set to the estimated values achieved through calibration.

minusEvals and plusEvals then adjust each parameter to determine
how a 10% increase or 10% decrease in parameter value will
affect the error values. The maximum deviation from ideal
error provides a measure of how sensitive the model is to that
parameter.
%}

%% Setup for Sensitivity Analysis

% clear cmd history for clarity
clc

% get the current directory
curdir = fileparts(which(mfilename));

% intialize storage vectors
[minusEvals,plusEvals,sensitivityVals] = deal([]);

parameters = setup; %run the setup function which creates the
```

```
%structure storing all variables necessary
%for evaluating the model (found in 'setup.m')

% store the values of the parameters in a vector
paramSet = parameters.ctrlParams;
paramVals = [paramSet.f0Vmax, paramSet.f0Km, paramSet.Vmax, paramSet.Km, ...
paramSet.K1, paramSet.p1, paramSet.p2, paramSet.p3, paramSet.Dh, ...
paramSet.cytcred];

% store +/- 10% values in new structures
paramMT = structfun(@(x)x*0.9,paramSet);
paramPT = structfun(@(x)x*1.1,paramSet);

% names of each parameters as they are stored
parameterIDs = {'Vmax','K1','Km','p1','p2','p3','f0Vmax','f0Km','Dh', ...
'cytcred'};

%% Evaluate E* and E*+/- 10%

% evalute E*, consistent across all parameter changes
[E_star,evaluations] = sensitivitySolver(parameters,paramSet,'Estar');
parameters.initialsOligo = evaluations{1}(60,:);
parameters.initialsFccp = evaluations{1}(135,:);
parameters.initialsInhibit = evaluations{1}(215,:);

% evaluate E* of plus and minus 10 for each parameter
for param=1:numel(parameterIDs)
parameterSet = paramSet;

% Change parameter to be evaluated at minus 10%
parameterSet.(parameterIDs{param}) = paramMT(param);
```

```
if param==10
parameters.Cytcred = paramMT(param);
parameterSet.Cytctot = parameterSet.cytcred + parameterSet.cytcox;
parameters.Cytctot = parameterSet.Cytctot;
end
minusEvals(param,1:5) = sensitivitySolver(parameters,parameterSet);

% Change parameter to be evaluated at plus 10%
parameterSet.(parameterIDs{param}) = paramPT(param);
if param==10
parameters.Cytcred = paramPT(param);
parameterSet.Cytctot = parameterSet.cytcred + parameterSet.cytcox;
parameters.Cytctot = parameterSet.Cytctot;
end
if param==9
plusEvals(param,1:5) = minusEvals(param,1:5);
else
plusEvals(param,1:5) = sensitivitySolver(parameters,parameterSet);
end

% store all the sensitivity vals in a matrix
for cond=1:5
sensitivityVals(param, cond) = max(abs(minusEvals(param,cond) ...
-E_star(cond))/(0.1*E_star(cond)),abs(plusEvals(param,cond) ...
-E_star(cond))/(0.1*E_star(cond)));
end

% calculate and display the sensitivity values
disp(['Sensitivity values for parameter ', parameterIDs{param}, ' are: ', ...
num2str(sensitivityVals(param,:))]);
end
```

```
% save results to a .mat and .txt file for viewing the sensitivity values
cd([curdir, '/SensitivityResults']); %change to Solutions folder
todayDate = date; %get the run date

% save the Best solution to the Solutions folder
resultsname = [todayDate '-SensitivityCoefficients'];
save(resultsname,'sensitivityVals');

disp(['Saving results to: ', resultsname]);
```

C.13 sensitivitySolver.m

```
function [errors,solutionEval] = sensitivitySolver(parameters,params,varargin)
%{
Created by: Chris Cadonic
=======================================
This function solves the full situation for my model by step-wise
solving the ODEs for each section using the appropriate equations.

Additionally, this function also solves for each condition in my model,
and is specifically made for the sensitivity analysis. Thus, the output
of this function is not the raw values, but instead the error values
calculated.
%}

%initialize variables
errors = [];

%Set the options for running ode45
```

```
options = odeset('NonNegative',[1,2,3,4]);

%format real data for 5 separate solutions
realData{1} = parameters.realo2Data;
realData{2} = realData{1}(1:parameters.oligoTime-1);
realData{3} = realData{1}(parameters.oligoTime: ...
parameters.fccpTime-1);
realData{4} = realData{1}(parameters.fccpTime: ...
parameters.inhibitTime-1);
realData{5} = realData{1}(parameters.inhibitTime:end);

%Solve by using ode for each section and passing along the final
%values as initial values for the next section
tic
[~,y1] = ode45(@baselineSystem, parameters.baselineTimes, ...
[parameters.Cytcred,parameters.O2,parameters.Hn, ...
parameters.Hp],options,params);
[~,y2] = ode45(@oligoSystem, parameters.oligoTimes, ...
[y1(end,1),y1(end,2),y1(end,3),y1(end,4)],options,params);
[~,y3] = ode45(@fccpSystem, parameters.fccpTimes, ...
[y2(end,1),y2(end,2),y2(end,3),y2(end,4)],options,params);
[~,y4] = ode45(@inhibitSystem, parameters.inhibitTimes, ...
[y3(end,1),y3(end,2),y3(end,3),y3(end,4)],options,params);

%store the first, solution for the entire model
solutionEval{1} = [y1;y2;y3;y4];

%if varargin in nonempty, then this is calculating Estar
if ~isempty(varargin)
parameters.initialsOligo = y1(end,:);
parameters.initialsFccp = y2(end,:);
```

```
parameters.initialsInhibit = y3(end,:);
end

%repeat solving the system for each section separately
[~,solutionEval{2}] = ode45(@baselineSystem, parameters.baselineTimes, ...
[parameters.Cytcred,parameters.O2,parameters.Hn, ...
parameters.Hp],options,params);
[~,solutionEval{3}] = ode45(@oligoSystem, parameters.oligoTimes, ...
parameters.initialsOligo,options,params);
[~,solutionEval{4}] = ode45(@fccpSystem, parameters.fccpTimes, ...
parameters.initialsFccp,options,params);
[~,solutionEval{5}] = ode45(@inhibitSystem, parameters.inhibitTimes, ...
parameters.initialsInhibit,options,params);

% loop over and calculate the error for each condition
for condition = 1:numel(solutionEval)
% calculate error for the entire model
errors(condition) = sum((realData{condition} ...
-solutionEval{condition}(:,2)).^2)/numel(realData{condition});
end
toc
```

Appendix D

Genetic Algorithm

D.1 General Description of a Genetic Algorithm

Calibration of the model was carried out by using an optimization algorithm to minimize the difference between model results and experimental results (see Section 2.2.7).

Calibration of the model for each segment was carried out using a genetic algorithm. The genetic algorithm is an optimization technique that borrows from natural genetic principles to guide how parameter values are varied during the optimization process. The general implementation of the genetic algorithm is as follows:

1. First create an initial population of n solvers.
2. Give each solver the initial m parameter values in the form $\{p1, p2, p3, ..., p_m\}$.
3. Mutate some of the solvers except for the very first solver, to create variation in the initial population.
4. Evaluate the objective functions for this population.
5. Check convergence criteria: If met, exit the GA and output best solver, if not, continue with GA.

6. Allow only the top x percent to survive toward the next population
7. Introduce breeding between these top solvers, with partial crossover between matching parameter values occurring until a new population is created (with parents + children).
8. Randomly introduce mutations in parameter values within the new population.
9. Re-evaluate objective functions for new population.
10. Repeat process from step 5 until convergence criteria is met.

The genetic algorithm utilized in the calibration of the model to experimental data is known as *Ferret*, as part of the *Qubist* MATLAB software package developed by Jason Fiege (Department of Physics, University of Manitoba). This is a very robust implementation of a genetic algorithm, incorporating more advanced features such as machine-learning algorithms altering the conditions of the genetic algorithm to adapt for more efficient optimization. For more information regarding this software, details are given on the website *http://www.nqube.com/qubist/*.

D.1.1 Objective Function

As discussed in section 2.2.7, the objective function used for optimizing the parameter values was the MSE function. This objective function is appropriate for parameter fitting in this model since there is only a single set of observations for oxygen concentration, thus a single output is compared to each matching experimental data point. A squared measure of the residual of the model provides a small remaining error as then the process of fitting the data becomes of order $O(n)$. More accurate estimates are also available, but then computational efficiency becomes a concern for evaluating large population sizes in the genetic algorithm. Thus, calibrating the model by minimizing MSE is an appropriate compromise for calibrating the model.

Bibliography

[1] D. Nelson and M. Cox. *Lehninger Principles of Biochemistry, Fourth Edition.* W.H. Freeman, 2004.

[2] I. Scheffler. *Mitochondria, Second Edition.* Wiley, 2007.

[3] CA. Mannella. Structure and dynamics of the mitochondrial inner membrane cristae. *Biochimica et Biophysica Acta*, 1763:542 – 548, 2006.

[4] A. Reeve, K. Krishnan, M. Duchen, and D. Turnbull, editors. *Mitochondrial Dysfunction in Neurodegenerative Disorders.* Springer, 2012.

[5] Berg. J., J. Tymoczko, and L. Stryker. *Biochemistry, Seventh Edition.* W.H. Freeman, 2010.

[6] M.T. Lin and M.F. Beal. Mitochondrial dysfunction and oxidative stress in neurodegenerative diseases. *Nature*, 442:787–795, 2006.

[7] P. Mishra and D. Chan. Mitochondrial dynamics and inheritance during cell division, development and disease. *Naure Reviews Molecular Cell Biology*, 15:634–636, 2014.

[8] P. Mitchell and J. Moyle. Chemiosmotic hypoethesis of oxidative phosphorylation. *Nature*, 213:137–139, 1967. doi: 10.1038/213137a0.

[9] N. Masaya, I. Takashi, A. Shinichi, H. Youichi, M. Hiroshi, S. Rika, and T. Kuniaki. Synthetic studies on oligomycins. synthesis of the oligomycin b spiroketal and polypropionate portions. *Bulletin of the Chemical Society of Japan*, 68:967–989, 1995.

[10] G. Ling. Oxidative phosphorylation and mitochondrial physiology: A critical review of chemiosmotic theory, and reinterpretation by the association-induction hypothesis. *Physiological Chemistry and Physics*, 13:29–96, 1981.

[11] M. Watabe and T. Nakaki. Mitochondrial complex i inhibitor rotenone inhibits and redistributes vesicular monamine transporter 2 via nitration in human dopaminergic sh-sy5y cells. *Molecular Pharmacology*, 74:933 – 940, 2008.

[12] N. Dairaku, K. Kato, K. Honda, T. Koike, K. Iijima, A. Imatani, H. Sekine, S. Ohara, H. Matsui, and T. Shimosegawa. Oligomycin and antimycin a prevent nitric oxide-induced apoptosis by blocking cytochrome c leakage. *The Journal of Laboratory and Clinical Medicine*, 143:143 – 151, 2004.

[13] Wu, F. and Yang, F. and Vinnakota, K.C. and Beard, D.A. Computer modeling of mitochondrial tricarboxylic acid cycle, oxidative phosphorylation, metabolite transport, and electrophysiology. *The Journal of biological chemistry*, 282:24525 – 24537, 2007. doi: 10.1074/jbc.M701024200.

[14] D. Beard. A biophysical model of the mitochondrial respiratory system and oxidative phosphorylation. *PLoS Computational Biology*, 1:252–264, 2005.

[15] B. Cortassa, S. and Aon, M. and Marban, E. and Winslow, R. and O'Rourke. An integrated model of cardiac mitochondrial energy metabolism and calcium dynamics. *Biophysical journal*, 84:2734 – 2755, 2003.

[16] S. Wei, A. and Aon, M. and O'Rourke, B. and Winslow, R. and Cortassa. Mitochondrial Energetics, pH regulation, and ion dynamics: A computational-experimental approach. *Biophysical journal*, 100:2894 – 2903, 2011.

[17] B. Korzeniewski, A. Noma, and S. Matsuoka. Regulation of oxidative phosphorylation in intact mammalian heart in vivo. *Biophysical Chemistry*, 116:145–157, 2005.

[18] B Korzeniewski and J. Zoladz. A model of oxidative phosphorylation in mammalian skeletal muscle. *Biophysical chemistry*, 92(1-2):17–34, 2001.

[19] S.P. Patel and S.S. Katyare. Differences in kinetic properties of cytochrome oxidase in mitochondria from rat tissues. A comparative study. *Verlag der Zeitschrift fr Naturforschung, Tbingen*, 60:785–791, 2005.

[20] J. Bazil, G. Buzzard, and A. Rundell. Modeling mitochondrial bioenergetics with integrated volume dynamics. *PLoS Computational Biology*, 6:1–16, 2010.

[21] M. Jafri, S.J. Dudycha, and B. O'Rourke. Cardiac energy metabolism: models of cellular respiration. *Annual Review of Biomedical Engineering*, 3:57 – 81, 2001.

[22] S. Kembro, J. and Aon, M. and Winslow, R. and O'Rourke, B. and Cortassa. Integrating mitochondrial energetics, redox and ROS metabolic networks: A two-compartment model. *Biophysical journal*, 104:332 – 343, 2013.

[23] G. Magnus and J. Keizer. Minimal model of β-cell mitochondrial Ca^{2+} handling. *American Journal of Physiology*, 273:C717 – C733, 1997.

[24] G. Magnus and J. Keizer. Model of β-cell mitochondrial calcium handling and electrical activity. I. Cytoplasmic variables. *American Journal of Physiology*, 274:C1158 – C1173, 1998.

[25] G. Magnus and J. Keizer. Model of β-cell mitochondrial calcium handling and electrical activity. II. Mitochondrial variables. *American Journal of Physiology*, 274:C1174 – C1184, 1998.

[26] R. Bertram, M.G. Pederson, D. Luciani, and A. Sherman. A simplified model for mitochondrial ATP production. *Journal of Theoretical Biology*, 243:575 – 586, 2006.

[27] H. Michel. The mechanism of proton pumping by cytochrome c oxidase. *Proceedings of the National Academy of Sciences*, 95:12819–12824, 1998.

[28] S. Jain and S. Nath. Kinetic model of atp synthase: ph dependence of the rate of atp synthesis. *Federation of European Biochemical Societies*, 476:113–117, 2000.

[29] J. Keener and J. Sneyd. *Mathematical Physiology I: Cellular Physiology, Second Edition.* Springer, 2009.

[30] C. Monge, N. Beraud, A.V. Kuznetsov, T. Rostovtseva, D. Sackett, U. Schlattner, M. Vendelin, and V.A. Saks. Regulation of respiration in brain mitochondria and synaptosomes: restrictions of adp diffusion in situ, roles of tubulin, and mitochondrial creatine kinase. *Molecular and Cellular Biochemistry*, 318:147–165, 2008. doi: 10.1007/s11010-008-9865-7.

[31] S. Serowy, S.M. Saparov, Y.N. Antonenko, W. Kozlovsky, V. Hagen, and P. Pohl. Structural proton diffusion along lipid bilayers. *Biophysical journal*, 84(2 Pt 1):1031–7, 2003. doi: 10.1016/S0006-3495(03)74919-4.

[32] N S Chandel, N S Chandel, G R S Budinger, G R S Budinger, P T Schumacker, and P T Schumacker. Molecular oxygen modulates cytochrome c oxidase functions. *The Journal of Biological Chemistry*, 271(31):18672–18677, 1996.

[33] D. Curti, M.C. Giangare, M.E. Redolfi, I. Fugaccia, and G. Benzi. Age-related modifications of cytochrome c oxidase activity in discrete brain regions. *Mechanisms of Ageing and Development*, 55:171–180, 1990.

[34] H James Harmon. Effect of age on kinetics and carbon monoxide binding to cytochrome oxidase in synaptic and non-synaptic brain mitochondria. *Mechanisms of Ageing and Development*, 53:35–48, 1990.

[35] D. Poburko, J. Santo-Domingo, and N. Demaurex. Dynamic regulation of the mitochondrial proton gradient during cytosolic calcium elevations. *The Journal of Biological Chemistry*, 286:11672–11684, 2011.

[36] J. Santo-Domingo and N. Demaurex. The renaissance of mitochondrial ph. *The Journal of General Physiology*, 139:415–423, 2012. doi: 10.1085/jgp.201110767.

[37] H. Rottenberg. The measurement of transmembrane electrochemical proton gradients. *Bioenergetics*, 7:61–74, 1975.

[38] M.T. Lin and M.F. Beal. The redox states of respiratory-chain components in rat-liver mitochondria ii. the "crossover" on the transition from state 3 to state 4. *Biochimica et Biophysica Acta*, 180:227–236, 1969.

[39] A. Sauerbeck, J. Pandya, I. Singh, K. Bittman, R. Readnower, G. Bing, and P. Sullivan. Analysis of regional brain mitochondrial bioenergetics and susceptibility to mitochondrial inhibition utilizing a microplate based system. *Journal of Neuroscience Methods*, 198, 2011.

[40] M. Brand and D. Nicholls. Assessing mitochondrial dysfunction in cells. *Biochemical Journal*, 435:297–312, 2011.

[41] D.F. Silva, J.E. Selfridge, J. Lu, L. E, N. Roy, L. Hutfles, J.M. Burns, E.K. Michaelis, S. Yan, S.M. Cardoso, and R. Swerdlow. Bioenergetic flux, mitochondrial mass and mitochondrial morphology dynamics in AD and MCI cybrid cell lines. *Human Molecular Genetics*, 22:3931–3946, 2013.

[42] WD. Parker Jr, J. Parks, CM. Filley, and Kleinschmidt-DeMasters BK. Electron transport chain defects in alzheimer's disease. *Neurology*, 44:1090–1096, 1994.

[43] V. Garcia-Escudero, P. Martin-Maestro, G. Perry, and J. Avila. Deconstructing mitochondrial dysfunction in alzheimer disease. *Oxidative Medicine and Cellular Longevity*, 2013:13 pages, 2013.

[44] R. Sterniczuk, R.H. Dyck, F.M. Laferla, and M.C. Antle. Characterization of the 3xtg-ad mouse model of alzheimer's disease: part 1. circadian changes. *Brain Research*, 1348:139 – 148, 2010.

[45] C. Pereira, MS. Santos, and C. Oliveira. Involvement of oxidative stress on the impairment of energy metabolism induced by a beta peptides on pc12 cells: protection by antioxidants. *Neurobiology of Disease*, 6:209 – 219, 1999.

[46] S. I. Rubinow. *Introduction to Mathematical Biology.* John Wiley and Sons, 1975.