# Methods for parameter identification in the Mitchell-Schaeffer model

by

Jacob Pearce-Lance

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Mathematics[1]

Department of Mathematics and Statistics
Faculty of Science
University of Ottawa



[1]The M.Sc. program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute of Mathematics and Statistics

# Abstract

This thesis focusses on the development and testing of optimization methods for parameter identification in cardiac electrophysiology models. Cardiac electrophysiology models are systems of differential equations representing the evolution of the trans-membrane potential of cardiac cells. The Mitchell-Schaeffer model is chosen for this thesis. The parameters included in the Mitchell-Schaeffer model are optimally adjusted so that the solution of the model has desired properties. Two optimization problems are formulated using least-square functions to identify parameters that match phase durations and parameters that fit entire potential recordings of swine heart tissue acquired via optical imaging techniques at different stimulation frequencies. The non-differentiable optimization methods (Compass Search and three other variants) are applied to solving both optimization problems for two reasons; First, the methods are studied to evaluate performance and second, the optimization process is evaluated to confirm its ability to identify parameters for the Mitchell-Schaeffer model.

# Résumé

Cette thèse se concentre sur le développement et l'analyse de méthodes d'optimisation pour l'identification de paramètres dans les modèles d'électrophysiologie cardiaque. Un modèle d'électrophysiologie cardiaque est un système d'équations différentielles qui représente l'évolution du potentiel transmembranaire de cellules cardiaques. Le modèle de Mitchell-Shaeffer sera utilisé pour cette thèse. Les paramètres inclus dans le modèle de Mitchell-Shaeffer sont ajustés optimalement pour que la solution du modèle satisfasse des propriétés désirées. Deux problèmes d'optimisation sont formulés en utilisant des fonctions de moindres carrés pour identifier des paramètres qui font correspondre les durées de phases et des paramètres qui font correspondre le potentiel prédit par le modèle avec celui provenant d'enregistrements faits sur des coeurs de cochons avec des techniques d'imagerie optique à différentes fréquences de stimulation. Des méthodes d'optimisation non différentiables (Compass Search et trois autres variantes) sont appliquées à la résolution des deux problèmes d'optimisation. Deux raisons motivent notre travail: 1) les méthodes sont étudiées pour évaluer leur performance et 2) le procédé d'optimisation est évalué pour confirmer sa capacité à identifier correctement les paramètres pour le modèle de Mitchell-Shaeffer.

**Acknowledgements**

First and foremost, I would like to thank my thesis supervisor Yves Bourgault for the guidance he has provided over the past two years. Our weekly meetings contributed greatly towards the progression of the research project and I am very happy to have had Yves as my supervisor, as he was very patient and pedagogical in introducing me to the academic world. I also thank Mihaela Pop and her team at the Sunnybrook research institute for providing me with the experimental data I used for my work. I thank all my colleagues also working with Yves for the small pointers and discussions on how things are done in this field. I thank the thesis defence examiners Dr. Dave Amundsen and Dr. Abdelaziz Beljadid for reading and evaluating my thesis. A big thanks goes out to my parents, Stéphane Lance and Stéphanie Pearce, for supporting me throughout my studies and without whom I would not be the person I am today. Finally, a special thank you is given to my girlfriend Émilie, who keeps me motivated and makes me blissful every day.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Heart disease is the second leading cause of death for Canadians [23]. Therefore, a lot of research is dedicated to understanding how the heart works and, by extension, what can hinder the heart from functioning normally. It is one thing to know what kind of pathologies the heart can succumb to, but it is also important to be able to identify said pathologies in patients. Diagnosis and treatment must be implemented in a reasonably fast manner so as to effectively heal or prevent damage from being done. Using mathematical models and methods, it is possible to visualize the electrical activity in the heart in order to understand or to predict healthy or pathological behaviour.

## 1.1 Electrophysiological context

This section briefly reviews cardiac electrophysiology concepts. References [8, 12, 31] provide a broader presentation of the subject. The heart is a large network of individual cardiac cells interacting with one another via electrical currents. The cell membrane acts as an isolating layer between the intra-cellular and extra-cellular domains, both of which are Ohmic conductors. This separation leads to a difference of electrical potential between the two domains, due to differences in the concentrations of the ions mostly responsible for the conduction of the electrical wave (sodium, calcium and potassium ions). The difference of potential, denoted $u$ and often referred to as the trans-membrane potential (the potential, for short), is by convention calculated as $u = u_i - u_e$, where $u_i$ is the intra-cellular potential and $u_e$ is the extra-cellular potential. In their natural state, cardiac cells have a "rest potential", which varies depending on the cell type. However, when subjected to an outside stimulation (i.e. an electrical current generated from a

difference of potential with neighbouring cells or by a stimulation electrode), the transmembrane potential increases rapidly before returning to its rest value. This process of increase-decrease is called an action potential (AP) and is characterized by four phases:

- Phase 1: Due to the external current, the flux of potential raising ions (primarily calcium and sodium) coming into the cell dominates the outward, potential diminishing flux (primarily potassium) which leads to a rapid increase of the transmembrane potential up to its maximal value. Ventricular cells, for example, have a resting value of roughly $-80$ mV which goes up to around 40 mV after phase 1 [8]. This phase is called "depolarization".

- Phase 2: In response to the previous unbalancing of inward and outward movement of ions, the outward flux increases to recreate equilibrium. The potential starts decaying back very slowly to its rest value, hence the name of this phase being "plateau".

- Phase 3: The outward current now dominates the inward current, which has been restored to its initial, non-excited rate, and so the potential drops more drastically than during phase 2, but not quite as fast as the increase in phase 1. By the end of this phase, potential will be very near rest value and the cell has become repolarized, giving phase 3 the name "repolarization".

- Phase 4: The channels through which the ionic fluxes pass are now inactive, so no further unbalancing of flows may occur. The cell is no longer susceptible to external current fluctuations and must recover before being able to begin a new depolarization-repolarization cycle. This phase is called "recovery".

In summary, when a region of the heart is sufficiently stimulated, each cell is excited in turn resulting in a rapid increase of the potential, a relatively long period of maintained elevated potential, a moderately fast decrease back to resting state and a relatively long recovery period. Since the APs of each cell are happening one after the other, there is a sort of "depolarization wave" that traverses the whole heart, normally starting from the sinoatrial node. The sinoatrial node is composed of self-excitable cells, usually called the "natural pacemaker" of the heart. The electrical wave starts from the sinoatrial node and is propagated to both atria, first right than left, causing them to contract (the function of the heart is to pump blood, so the increase of potential causes the necessary contraction). The wave continues through the atria to reach the atrioventricular node,

which is the relay between the atria and the ventricles. Elsewhere, the atria and ventricles are separated by a non-conductive layer. The atrioventricular node conducts the current rather slowly from one medium to the other which causes a delay between the contraction of the atria and that of the ventricles, allowing the latter to properly fill up with blood. Once through this node, the current follows the Purkinje fibres that run all through the ventricles, leading to the depolarization and contraction of the right and left ventricles, completing the entire heartbeat.

It is worth noting that there is a variety of different cardiac cells, which react differently to stimulating currents. Hence, the AP described above, although accurately representing the general shape of APs for cardiac cells, can look a bit different depending on where the cell is found in the heart (e.g. the proportions between the durations of the phases will differ from cell type to cell type). The differences in shape and delays between activations of the APs of each individual cell are all synchronized together to assure that the heart beats effectively and consistently. This also means that irregularities or "unexpected" behaviours of different heart regions can cause irregular conduction patterns that lead to pathological heart conditions.

## 1.2 Mathematical models

Over the years, electrophysiologists have developed ways of recreating experimental data in order to predict changes or evolution of the electrical activity in the heart. Mathematical models are systems of differential equations (DEs) putting physical quantities in relation with each other. For cardiac electrophysiology models, the trans-membrane potential $u$, or more precisely its time derivative $\frac{du}{dt}$, is the main property represented. There are many different models that have been created to predict the shape of the AP, with each model having its own intrinsic properties. Ionic models characterize the electrical activity of the heart by considering the main flows of ions in and out of the cell. There are two categories of ionic models: physiological models and phenomenological models. Physiological models typically include many variables and consider the ionic currents and concentrations to represent the underlying physiology. The Hodgkin-Huxley model [10] and the Beeler-Reuter model [2] are examples of physiological models. Phenomenological models are simpler models used to reproduce the macroscopic behaviour of the cell, including the evolution of the trans-membrane potential and one or two variables representing globally the "slow" behaviour of ions. The Fitzhugh-Nagumo model [6, 17] and the Mitchell-Schaeffer model [16] are examples of phenomenological models.

The model studied throughout this thesis is the Mitchell-Schaeffer model, introduced by Colleen Mitchell and David Schaeffer in 2003. A precise statement of the model is found in section 2.1.

The models usually have many adjustable parameters which can affect the predicted AP. The adjustment of these parameters becomes important when trying to personalize the models using medical data, as is done in [26, 27]. Varying the parameters affects the predicted AP in many ways and often this effect is difficult to characterize, especially when many parameters vary simultaneously. Some simpler models, such as the Mitchell-Schaeffer model, may be analyzed using asymptotic methods to connect the parameters to some properties, such as the phase durations [16, 26, 29]. However, addressing parameter adjustment in fully non-linear models is not as easy, especially when complex properties such as restitution properties must be observed. Some work has been done on this subject, such as the use of simulated annealing to compare ionic models [14] and the use of genetic algorithms to build a cell-specific electrophysiology model [9] or to adjust conductances among other parameters in non-linear models using direct voltage recordings [11, 32]. The work done in [32] resembles the work done in this thesis as recorded APs are included in a least-square function to match restitution properties with a model. However, this thesis strays away from that work by exploring different optimization methods (i.e. non genetic methods), by studying a different model and by utilizing different cost functions, both in the formulation and type of data input to the functions.

Another work [24] that closely relates to this one makes use of the Nelder-Mead optimization method [18] to adjust parameters in the Mitchell-Schaeffer model to match phase durations and to fit a single AP recording. This thesis explores different optimization methods in search of a better alternative to the Nelder-Mead algorithm while also extending the model-to-data fitting process to multiple AP recordings to account for restitution properties.

## 1.3 Scope of the thesis

The work done in this thesis is separated into four chapters. The second chapter presents the Mitchell-Schaeffer model and the numerical methods used throughout the thesis. The model is recalled from previous works and some of its properties are recalled and clarified. The numerical methods used to solve the model and to analyze its solution are presented in detail. These methods are implemented in the open-source computation software Octave [20] (version 4.2.2), which is used for all computations throughout the thesis.

The third chapter introduces two optimization problems and presents the optimization methods that are developed to solve them. The optimization problems are designed to help identify parameters for which the chosen mathematical model predicts certain properties for the trans-membrane potential, ranging from phase durations to the full shape of the APs and restitution properties. The optimization methods, of which one is standard and three others are our original variants, are presented along with theoretical concepts that support their validity, such as a proof of convergence.

The fourth chapter consists of an analysis, using numerical test cases, of the performance of the previously mentioned optimization methods. One of the optimization problems is used to study how the algorithms perform compared to each other and how the algorithms perform under certain modifications, such as changing the algorithm parameters.

The fifth chapter also contains an analysis using numerical tests of the performance of the optimization methods, but this time in a more practical setting. Experimental data obtained on pigs at the Sunnybrook Research Institute are reformatted so that the optimization process can fit the model to real-life data. Many different scenarios are considered, such as fitting the model to multiple recordings at once, to validate this optimization process.

# Chapter 2

# Mathematical background

This chapter presents the mathematical background for the thesis. Sections 2.1 and 2.2 introduce the mathematical model used for this work and the behaviour of its solution, respectively. The numerical methods used to compute this solution and other useful properties are exhibited in section 2.3.

## 2.1 The Mitchell-Schaeffer model

The work done in this thesis focuses on the Mitchell-Schaeffer model [16], which describes the electrical activity of a single cell. It is a system of two ODEs that relate to two variables, namely the difference of potential across the cell membrane and what is called a "gating variable", denoted by $u$ and $v$, respectively. The MS model reads as

$$\frac{du}{dt} = f(u,v) + I_{stim}(t), \text{ with } f(u,v) = -\frac{1}{\tau_{in}} v u^2 (u-1) - \frac{1}{\tau_{out}} u, \tag{2.1}$$

$$\frac{dv}{dt} = g(u,v), \text{ with } g(u,v) = \begin{cases} \frac{1-v}{\tau_{open}} & \text{if } u < u_{gate}, \\ \frac{-v}{\tau_{close}} & \text{if } u \geq u_{gate}. \end{cases} \tag{2.2}$$

Both $u$ and $v$ are functions of time. The gating variable represents, in a way, how "open" the ionic channels of the cell membrane are and how enabled the inward flux of ions is. A high value of $v$ means that the ions are coming into the cell quite easily while a low value means that the exterior ions are somewhat denied passage through the cell membrane. In this model, $0 \leq u \leq 1$, $0 \leq v \leq 1$ , with $t \geq 0$ in ms. The variable $u$ is a rescaled trans-membrane potential which should normally vary so that the AP has an amplitude of approximately 100 mV [12]. As seen in section 1.1, there are mainly

three currents that affect $u$: the inward currents of $Ca^{2+}$ and $Na^{+}$ ions and the outward current of $K^{+}$ ions. Calling the sum of the inward currents $I_{in}$ and the outward current $I_{out}$, $f$ can be rewritten $f(u,v) = I_{in} + I_{out}$ with $I_{in} = -\frac{1}{\tau_{in}} v u^2 (u-1)$ and $I_{out} = -\frac{1}{\tau_{out}} u$. The other term in equation (2.1), $I_{stim} = I_{stim}(t)$, is the stimulating current, which would come from either a neighbouring cell or a stimulation electrode. The choice of $I_{stim}$ is discussed in section 2.2. The function $g$ has a discontinuity at $u = u_{gate}$: it is positive when $u < u_{gate}$, meaning $v$ is increasing and it is negative when $u \geq u_{gate}$, meaning $v$ is decreasing. The effect of this discontinuity on the regularity of $u$ and $v$ is taken into account when working with this model, as mentioned in section 3.2. The model involves 5 parameters: $\tau_{in}$, $\tau_{out}$, $\tau_{open}$, $\tau_{close}$ and $u_{gate}$. Roughly, each $\tau_i$ affects the time scale of a different phase of the AP. $\tau_{in}$ mostly governs the duration of phase 1, $\tau_{close}$ phase 2, $\tau_{out}$ phase 3 and $\tau_{open}$ phase 4. $u_{gate}$ however does not affect AP significantly, it is controlling the excitability threshold of the cell. The influence of each of the parameters on the evolution of the AP is analyzed in more detail in [24]. Typical values for each of these five parameters are [16, 24]

$$\tau_{in} = 0.3,\ \tau_{out} = 6,\ \tau_{open} = 130,\ \tau_{close} = 150 \text{ and } u_{gate} = 0.13.$$

To find solutions of (2.1)-(2.2) numerically and plot them like in figure 2.1, an initial condition is needed. Unless stated otherwise, the equilibrium state is used as the initial condition:

$$[u(0), v(0)] = [0, 1]. \tag{2.3}$$

## 2.2 Behaviour of the solution

In order to analyze the behaviour of the solution of (2.1)-(2.3), similar steps as in [16] are taken. First assume that $\tau_{in} \ll \tau_{out} \ll \tau_{open},\ \tau_{close}$. Then, assuming $I_{stim}(t) \equiv 0$, the nullclines are given by:

$$f(u,v) = 0 \text{ and } g(u,v) = 0, \text{ i.e.}$$

$$-\frac{1}{\tau_{in}} v u^2 (u-1) - \frac{1}{\tau_{out}} u = 0 \text{ and } 0 = \begin{cases} \frac{1-v}{\tau_{open}} & \text{if } u < u_{gate}, \\ \frac{-v}{\tau_{close}} & \text{if } u \geq u_{gate}. \end{cases}$$

From the first equation, if $u \neq 0$, $v$ is expressed as a function of $u$

$$v = \frac{\tau_{in}}{\tau_{out}} \frac{1}{u(1-u)} \tag{2.4}$$

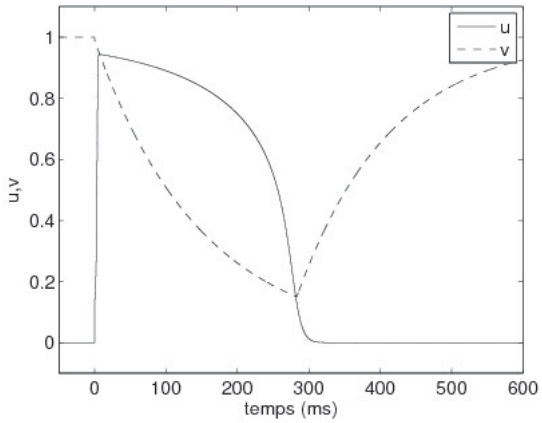



Figure 2.1: Typical profile of solutions $u$ and $v$ of (2.1)-(2.3) plotted in time (taken from [28]).

or $u$ as a function of $v$

$$u = u_{\pm} = \frac{1}{2} \pm \sqrt{\frac{1}{4} - \frac{\tau_{in}}{\tau_{out} v}} \tag{2.5}$$

and from the second equation,

$$v = \begin{cases} 1 & \text{if } u < u_{gate}, \\ 0 & \text{if } u \geq u_{gate}. \end{cases} \tag{2.6}$$

Equations (2.4), (2.6) and the equation $u = 0$ describe the nullclines in the $u$-$v$ plane. The AP and phase portrait in figures 2.1 and 2.2, respectively, show what is going on. Starting from the equilibrium state in (2.3), stimulus causes $u$ to increase while $v$ changes very little, so the trajectory of the solution moves horizontally to the nullcline (2.4). This corresponds to phase 1 of the AP. Since $u$ has now surpassed $u_{gate}$, $v$ starts to decrease and the trajectory follows the nullcline downwards. This is phase 2. When $v$ reaches its minimal value of $4\frac{\tau_{in}}{\tau_{out}}$ (minimum that can be found by minimizing (2.4) over the interval $(0, 1)$), $u$ starts decreasing more rapidly. The trajectory "falls off" the nullcline and continues left, representing phase 3. Finally, since $u$ is now smaller than $u_{gate}$ again, $v$ rises again and the solution returns towards its equilibrium, signaling that the cell may be excited again, which corresponds to phase 4.

Since the system starts at an equilibrium state, external stimulus is needed in order to observe an AP. $I_{stim}(t)$ plays this role, hence it is important to specify how $I_{stim}$ is chosen. Near the equilibrium point, $I_{in}$ and $I_{out}$ are essentially zero and the change in $v$

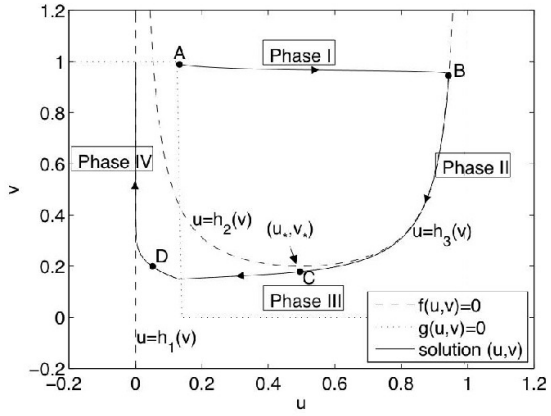



Figure 2.2: The phase diagram of $u$ and $v$, with the nullclines in dashed or dotted lines and each phase of the AP associated to a section of the phase diagram. $u = h_1(v) = 0$ is a nullcline, $u = h_2(v)$ and $u = h_3(v)$ are the $u_-$ and $u_+$ of (2.5), respectively, and the dotted line is the nullcline given in (2.6). Points A, B, C and D roughly represent the start of each phase (taken from [28]).

is negligible. So (2.1) can be seen as a separable ODE and solved to get

$$u(t) \approx \int_0^t I_{stim}(r)dr. \tag{2.7}$$

If $I_{stim}$ is a piecewise constant function (i.e. zero everywhere except on a certain intervals where it is constant and non-zero), then (2.7) becomes

$$u(\Delta t) \approx \Delta t \cdot A,$$

where $\Delta t$ is the length of a short time interval near $t = 0$ when $I_{stim}$ is non-zero and $A$ is the amplitude of the stimulus. If $\Delta t$ is fixed, then $A$ must be large enough so that after $\Delta t$, $u$ has crossed the nullcline (2.5) and an AP is initiated. So

$$u(\Delta t) \approx \Delta t \cdot A \geq \frac{1}{2} - \sqrt{\frac{1}{4} - \frac{\tau_{in}}{\tau_{out} v}} = u_-,$$

which gives

$$A \geq \frac{1}{\Delta t}\left(\frac{1}{2} - \sqrt{\frac{1}{4} - \frac{\tau_{in}}{\tau_{out} v}}\right). \tag{2.8}$$

In practice, to ensure that (2.8) is satisfied, it suffices to choose some safety factor $\beta > 0$ ($\beta$ is small as to not overstimulate) and to set

$$A = \frac{1}{\Delta t}\left(\frac{1}{2} - \sqrt{\frac{1}{4} - \frac{\tau_{in}}{\tau_{out} v^*}}\right) \cdot (1 + \beta), \tag{2.9}$$

with $v^*$ being the maximum of $v$ near the time of stimulation (e.g. $v^* = 1$ at $t = 0$). $I_{stim}$ is then written

$$I_{stim}(t) = \begin{cases} A & \text{if } t \in [n \cdot BCL, n \cdot BCL + \Delta t],\ n = 0, 1, 2, \dots \\ 0 & \text{otherwise,} \end{cases} \tag{2.10}$$

where BCL is the Basic Cycle Length (i.e. the delay between stimulations). BCL is specified when results are given. By restricting $n$ to only 0, $I_{stim}$ only stimulates once and a single AP is observed.

### 2.2.1 Restitution

When $I_{stim}$ is defined as in equation (2.10) with no restriction on $n$, multiple consecutive AP can be observed. Depending on the values of the parameters $\tau = [\tau_{in}, \tau_{out}, \tau_{open}, \tau_{close}]$ and the basic cycle length BCL, different behaviours are possible. The different behaviours and the conditions under which they occur are explored more deeply in [16]. Recalled here are only a few notions from that paper that are of interest in the context of the work presented in later sections.

"Restitution properties" is a catchall term referring to the behaviour of the model subject to multiple stimulations. Examples of restitution properties are whether or not each stimulation induces an AP, the relative length of the different phases of the APs and the relative "peak" values of the APs when the BCL is varied. Peak values are the values of the local maxima of $u$.

When observing the solution of the model for multiple AP, as in figure 2.3, it is noticeable that the first AP in particular differs from the rest. It peaks at a higher value and last a bit longer than the others. This is because the first stimulation is applied at equilibrium (2.3), while the following stimulations occur before the system is back to equilibrium. This is seen by the fact that $v$ has not reached its equilibrium value 1 when the following stimulation is applied. In fact, $v = v^*$ as defined in section 2.2. Since $v$ starts at a smaller value for the subsequent AP, the inward current $I_{in}$ in (2.1) is smaller throughout the AP and $u$ starts decreasing sooner, so $u$ can not reach as high a value as in the first AP. In general, the larger the value of $v$ at the time of stimulation, the higher the potential $u$ peaks. Usually, the first few APs differ a bit before the solution stabilizes into a periodic pattern where every AP are essentially the same (peak value and duration), but the most notably different AP is often the first. Even though the APs in figure 2.3 do not seem too different, the difference between the APs is accentuated

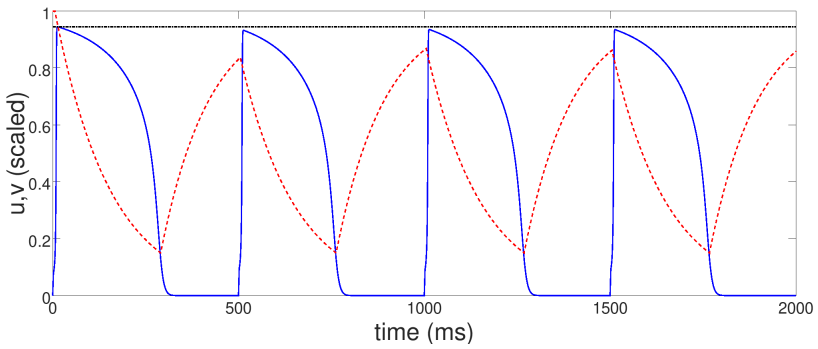



Figure 2.3: The solution of the MS model with $\tau = [0.3, 6, 130, 150]$ stimulated every 500 ms, where $u$ is the solid blue line, $v$ is the dashed red line. The black dash-dotted line is drawn at the peak of the first AP to show that the following AP have smaller amplitudes.

when BCL gets smaller.

This last remark leads to another notion that counts as a restitution property. Reducing BCL can lead to a shift in the behaviour of the solution. In figure 2.3, there is an AP every 500 ms, so every stimulation induces an AP. In particular, the solution seems to have fallen into a stable pattern as soon as the second AP. As in [16], this is called 1:1 response. However, 1:1 response does not happen for all stimulation periods BCL. For smaller BCL, it is possible that only every other stimulation induces an AP. If a stimulation is applied too soon, it is possible that $v$ has not yet increased enough for the cell to be excitable again and no AP is observed for that stimulation. Figure 2.4 illustrates this phenomenon. After a stimulation that fails to induce an AP, $v$ continues to increase and in all likelihood the next stimulation will come at a time where $v$ is now large enough to permit an AP to happen. This is known as 2:1 response [16]. Of course, it can happen that BCL is so small that multiple consecutive stimulations fails to induce an AP, leading to what could perhaps be called n:1 response, but the 2:1 response would be observed before, so this case is not considered. Finally, the MS model exhibits a third kind of behaviour, known as 2:2 response [16]. This happens when each stimulation induces an AP, but there is an alternation between shorter APs and longer ones. Figures 2.5 and 2.6 show this behaviour. In these figures, there is an alternation between APs lasting approximately 307 ms and others lasting approximately 296 ms. Also, the peaks

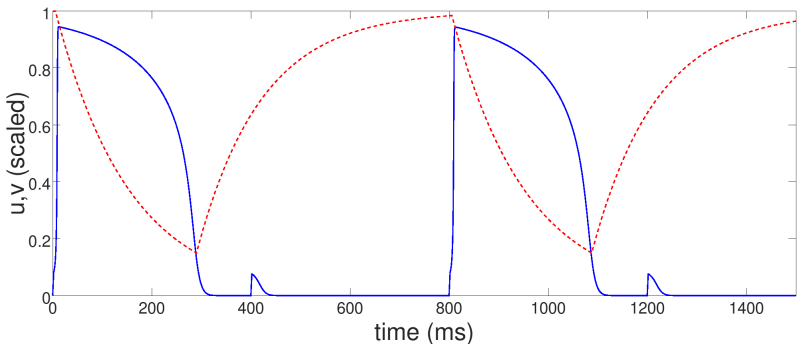



Figure 2.4: The solution of the MS model ($u$ in blue, solid, $v$ in red, dashed) with $\tau = [0.3, 6, 130, 150]$ stimulated every 400 ms.

alternate between approximately 0.907 for longer APs and 0.901 for shorter APs.

Along these lines, the final restitution property of interest here can explain in part why some APs would be shorter than others. Since the peak potential and duration of an AP depend on the value $v^*$ at which the gating variable $v$ recovered after a previous AP, there is a relation between the duration of the AP and the duration of the previous recovery period, also called the diastolic interval DI. Without going into much details (more are found in [16]), shorter DI lead to shorter AP. A graph of this relation as in figure 2.7 is called a restitution curve. For this particular restitution curve, the duration of the ninth AP (APD), which is the sum of durations of phases 1, 2 and 3, is plotted against the duration of the eight DI, which is the duration between the end of phase 3 of the eight AP and the beginning of phase 1 of the ninth AP. The way the duration of the phases is found is explained in section 2.3. The ninth AP is considered so that the system has had time to stabilize after the first few stimulations. Each pair (APD,DI) for a certain BCL gives one point on the restitution curve, so the BCL is gradually reduced to get multiple points.

## 2.3 Numerical methods

Since the MS model cannot be solved analytically, numerical approximations of the solution are used. The following section details the numerical methods used to solve the MS model, explains how phase durations are extracted from numerical solutions and exhibits a method of polynomial approximation that can be applied on the solution.

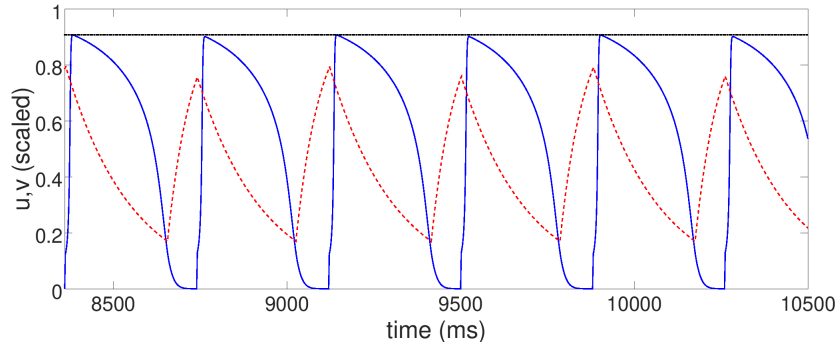



Figure 2.5: The solution of the MS model ($u$ in blue, $v$ in red) with $\tau = [0.6, 10, 70, 190]$ stimulated every 380 ms. The black line is the peak of the highest AP.

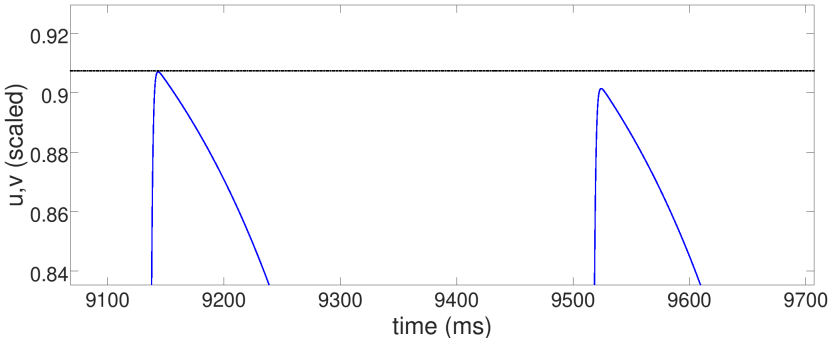



Figure 2.6: Zoomed version of figure 2.5 to emphasize that consecutive APs have slightly different peak values.

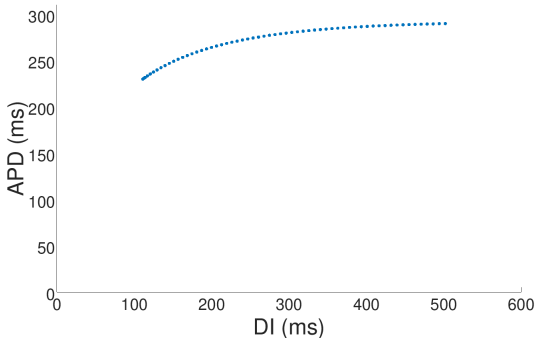



Figure 2.7: Restitution curve for MS model with parameters $\tau = [0.3, 6, 130, 150]$ starting at BCL= 800

### 2.3.1 Solution of the model

The first step is solving (2.1)-(2.3) given a set of parameters $\tau = [\tau_{in}, \tau_{out}, \tau_{open}, \tau_{close}]$. This is done using the computing software Octave, which shares many similarities with MATLAB. Within Octave, the ODE solver *ode45* is a function that solves ODEs using an explicit Dormand-Prince method of order 4 [21]. Although a quick description of this method follows, one can consult [5, 31] for more ample details. This method is a type of Runge-Kutta method. Consider the following ODE

$$\begin{cases} \frac{dy}{dt} = f(y,t), \ t > t_0 \\ y(t_0) = y_0. \end{cases} \tag{2.11}$$

Assuming the value $y(t_n)$ is known, the ODE (2.11) can be integrated to get

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(y,t)\ dt. \tag{2.12}$$

Most of the time, the integral term in (2.12) can not be calculated exactly, so an approximation is needed. For Runge-Kutta methods, the approximation is made by computing a certain number of intermediate values of $f(y,t)$ for $t \in [t_n, t_{n+1}]$ and using a weighted sum of these:

$$\int_{t_n}^{t_{n+1}} f(y,t)\ dt \approx \Delta t \sum_{i=1}^{s} b_i k_i, \tag{2.13}$$

where $\Delta t = t_{n+1} - t_n$, $s$ is the chosen number of stages, $k_i$ are the intermediate values of $f(y,t)$, which are calculated using

$$k_i = f(t_n + c_i \Delta t, y(t_n) + \Delta t \sum_{i=1}^{s} a_{ij} k_j) \text{ for } i = 1, 2, \ldots, s, \tag{2.14}$$

and $a_{ij}$, $b_i$, $c_i$ are specific to each method. For the Dormand-Prince method of order 4 implemented in *ode45*, these coefficients are listed in table 2.1 (using the first row $b_i$) [5]. So equations (2.12)-(2.14) give a way of approximating the value of $y$ one time step further, but it must be ensured that this approximation is accurate enough. To do this, the method also calculates a fifth order approximation using the coefficients in table 2.1 (using the second row $\hat{b}_i$) and uses the difference between both solutions as an error estimate $err$ for the fourth order solution. If $err \leq tol$, with $tol$ a chosen tolerance, the calculation is considered a success and the method can proceed to $y(t_{n+2})$. If the error is too large, then the calculation has failed and a smaller step-size must be taken in the

Table 2.1: Butcher tableau of coefficients for fourth ($b_i$) and fifth ($\hat{b}_i$) order Dormand-Prince method

| $c_i$ | $a_{ij}$ | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| $\frac{1}{5}$ | $\frac{1}{5}$ | 0 | 0 | 0 | 0 | 0 | |
| $\frac{3}{10}$ | $\frac{3}{40}$ | $\frac{9}{40}$ | 0 | 0 | 0 | 0 | |
| $\frac{4}{5}$ | $\frac{44}{45}$ | $-\frac{56}{15}$ | $\frac{32}{9}$ | 0 | 0 | 0 | |
| $\frac{8}{9}$ | $\frac{19372}{6561}$ | $-\frac{25360}{2187}$ | $\frac{64448}{6561}$ | $-\frac{212}{729}$ | 0 | 0 | |
| 1 | $\frac{9017}{3168}$ | $-\frac{355}{33}$ | $\frac{46732}{5247}$ | $\frac{49}{176}$ | $-\frac{5103}{18656}$ | 0 | |
| 1 | $\frac{35}{384}$ | 0 | $\frac{500}{1113}$ | $\frac{125}{192}$ | $-\frac{2187}{6784}$ | $\frac{11}{84}$ | |
| $b_i$ | $\frac{5179}{57600}$ | 0 | $\frac{7571}{16695}$ | $\frac{393}{640}$ | $-\frac{92097}{339200}$ | $\frac{187}{2100}$ | $\frac{1}{40}$ |
| $\hat{b}_i$ | $\frac{35}{384}$ | 0 | $\frac{500}{1113}$ | $\frac{125}{192}$ | $-\frac{2187}{6784}$ | $\frac{11}{84}$ | 0 |

hope of satisfying the error tolerance. The step-size $\Delta t$ is adjusted after each calculation, success or failure. This step-size adjustment is called adaptive step-size control and is done using the *integrate_adaptive* subroutine in Octave [7], which uses the formula

$$\Delta t \leftarrow \Delta t \left( \frac{tol}{err} \right)^{\frac{1}{5}}. \tag{2.15}$$

Observing equation (2.15) confirms that the step-size gets smaller after failed steps ($\frac{tol}{err} < 1$) and gets larger after successful ones ($\frac{tol}{err} \geq 1$).

The *ode45* routine is applied to the system (2.1)+(2.2) with initial condition (2.3). Additional options are specified in the input of *ode45* to assure the accuracy of the solution [21, 30]. "$MaxStep$" is the maximal step length allowed (if an adjustment would make the step-size larger, this value is chosen instead). "$InitialStep$" is the size of the first step. "$RelTol$" is used to calculate the above mentioned error tolerance that a step must satisfy in order to be successful. "$AbsTol$" is also a tolerance for the error. In short, the error $err$ must be smaller than $tol = \max(RelTol \cdot |y(t_{n+1})|, AbsTol)$ [15]. These parameters are all changed from their default values to values given in section 4.1. The solution is calculated on the interval $[0, T]$ for a chosen value of $T$. The output is a set of $t_j \in [0, T]$ and the values of $u$ and $v$ at those $t$ values. It is worth noting that the value of $u$ or $v$ for any other $t$ is not given.

### 2.3.2 Phase durations

Once the solution of (2.1)-(2.3) is computed, the duration of each of the four phases can be calculated. Phase durations are calculated based on when $u$ and $v$ cross certain thresholds. These are denoted by $\gamma_i$, for $i = 1, 2, 3, 4, 5$. If only one AP is considered (there is no periodic stimulation), the four phases may be defined in this way:

- Since $u$ starts at 0 and increases, phase 1 starts when $u$ reaches $\gamma_1$ and ends when it reaches $\gamma_2$.
- Phase 2 then starts and lasts until $u$ decreases to $\gamma_3$.
- Phase 3 then starts and lasts until $u$, still decreasing, reaches $\gamma_4$.
- Phase 4 then starts and lasts until $v$, which had gone down from its rest value of 1 but is now increasing again, reaches $\gamma_5$.

Figure 2.8 illustrates this definition of the four phases. The times at which these events happen must be determined using the output of *ode45*. In all likelihood, $u(t_j) \neq \gamma_j$ for any $t_j$ given by *ode45*. However, there are two consecutive values of $t$, say $t_{j-1}$ and $t_j$ for which $u(t_{j-1}) \leq \gamma_i$ and $u(t_j) > \gamma_i$, or vice-versa. This means that the time $T_i$ at which $u$ crosses $\gamma_i$ (i.e. $u(T_i) = \gamma_i$) is in $[t_{j-1}, t_j)$. Once such an interval is found, linear interpolation is used to find $T_i$:

$$\begin{cases} T_i & = t_{j-1} + \frac{t_j - t_{j-1}}{u(t_j) - u(t_{j-1})}(\gamma_i - u(t_{j-1})), \text{ with } i = 1, 3, 4, \\ T_2 & = \arg \max_{t \in [t_{j-1}, t_j)} u(t). \end{cases} \tag{2.16}$$

The first formula also applies to finding $T_5$, which is the time when $v$ crosses $\gamma_5$ (replace $u$ by $v$ and $\gamma_i$ by $\gamma_5$). Hence, finding $T_1$ through $T_5$ consists of going through the array of times $t_j$ given by *ode45* to identify the five intervals on which to do linear interpolation using (2.16). In particular, $T_i < T_{i+1}$. Calling $P_i$ the length of phase $i$,

$$P_i = T_{i+1} - T_i, \quad i = 1, 2, 3, 4, \tag{2.17}$$

gives the length of each phase.

Since phases 1 through 4 are meant to represent the different periods of the AP, it is necessary to choose the $\gamma_i$ meaningfully. The parameter $\gamma_1$ is set to $u_{gate}$, so phase 1 starts when $u$ has reached the threshold potential for depolarization. The parameter $\gamma_2$

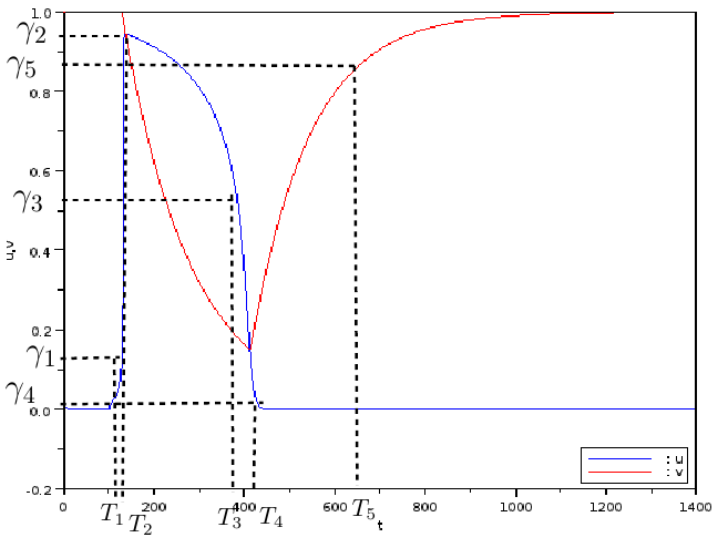



Figure 2.8: Solutions $u$ and $v$ of (2.1)-(2.3) with thresholds $\gamma_1$ through $\gamma_5$ and times $T_1$ through $T_5$ at which the solution cross the thresholds (taken from [24]).

is set equal to the local maximum around the peak of the AP, so that phase 2 starts when $u$ has stopped increasing. This means that when going through the process described above to determine $T_2$, it is necessary to find the first time interval following $T_1$ on which $u$ is smaller at the end than at the beginning of the interval. The parameter $\gamma_3$ is an intermediate value for which it is considered that the plateau phase is over and repolarization has begun. The parameter $\gamma_4$ is a small value for which $u$ is considered close enough to its resting value of 0. Finally, The parameter $\gamma_5$ is chosen so that the cell is seen as "excitable" again when $v$ crosses this value (i.e. the channels are open enough for depolarization to start again). Unless specified otherwise, the following values of $\gamma_i$ are used for this work

$$[\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5] = [0.13, \gamma_2, 0.5, 0.05, 0.9].$$

The parameter $\gamma_2$ has a variable value since the local max of $u$ can change slightly depending on the parameters and the stimulating current, but usually $\gamma_2 \approx 0.95$ for a single AP.

In the case that phase durations are needed for the solution of (2.1)-(2.3) stimulated periodically, the same calculations are done to identify $T_1$, $T_2$, $T_3$ and $T_4$ for each AP, denoting by $T_{i,j}$ the time $u$ crosses the $i$th threshold of the $j$th AP. Using similar notation,

the phase durations $P_{i,j}$ are obtained via

$$\begin{cases} P_{i,j} &= T_{i+1,j} - T_{i,j}, \quad i = 1, 2, 3 \ , j = 1, 2, \ldots, \\ P_{4,j} &= T_{1,j+1} - T_{4,j}, \quad j = 1, 2, \ldots. \end{cases}$$

The time $T_5$ is no longer considered because there is no way of guaranteeing that $v$ crosses the threshold $\gamma_5$ for reasons explained in section 2.2.1. The phase duration $P_{4,j}$ is the duration of the diastolic interval (DI) of the $j$th AP or $j$th diastolic interval.

### 2.3.3 Cubic Spline

As explained in section 2.3.1, the solution of the model is computed for specific points $t_j$. However, it will be necessary in certain scenarios to have $u(t)$ for values of $t \neq t_j$ for all values of $j$. The method described in this section is called cubic spline interpolation with not-a-knot boundary conditions [1] and it is implemented in Octave using the *spline* routine [22]. The method is used to find a continuous function that will approximate $u(t)$ for all values $t \in [0, T]$.

Let $(x_0, y_0)$,...,$(x_n, y_n)$ be a set of $n+1$ points in $\mathbb{R}^2$, called "knots", with $a = x_0 < x_1 < \ldots < x_n = b$. The goal of cubic spline interpolation is to find a function $s \in C^2[a, b]$ which is a piece-wise cubic polynomial $s_i$ on each interval $[x_i, x_{i+1}]$, for $i = 0, \ldots, n-1$ such that $s(x_i) = y_i$ for $i = 0, \ldots, n$.

To construct the function $s$, define the cubic pieces $s_i$ as

$$s_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \text{ for } i = 0, \ldots, n-1. \tag{2.18}$$

By identifying the $4n$ coefficients $a_i$, $b_i$, $c_i$ and $d_i$, $s$ is completely determined by

$$s(x) = s_i(x) \text{ if } x \in [x_i, x_{i+1}].$$

The following conditions are set on the $s_i$:

$$s_i(x_i) = y_i \text{ for } i = 0, \ldots, n-1, \tag{2.19}$$

$$s_{n-1}(x_n) = y_n, \tag{2.20}$$

$$s_i(x_{i+1}) = s_{i+1}(x_{i+1}) \text{ for } i = 0, \ldots, n-2, \tag{2.21}$$

$$s_i'(x_{i+1}) = s_{i+1}'(x_{i+1}) \text{ for } i = 0, \ldots, n-2, \tag{2.22}$$

and

$$s_i''(x_{i+1}) = s_{i+1}''(x_{i+1}) \text{ for } i = 0, \ldots, n-2. \tag{2.23}$$

Conditions (2.19) and (2.20) ensure that $s$ passes through each of the knots. Conditions (2.21), (2.22) and (2.23) ensure that $s \in C^2[a,b]$ by forcing the $s_i$ and their derivatives to match at the knots. There are now $4n-2$ equations for $4n$ unknowns, so two more conditions are required. The way of choosing these two extra conditions varies from method to method. Octave's *spline* implements the not-a-knot conditions, which are

$$s_0'''(x_1) = s_1'''(x_1) \text{ and } s_{n-2}'''(x_{n-1}) = s_{n-1}'''(x_{n-1}). \tag{2.24}$$

The not-a-knot conditions (2.24) essentially force the $x_1$ and $x_{n-1}$ knots to not really be knots after all, since $s_0 \equiv s_1$ and $s_{n-2} \equiv s_{n-1}$ under these conditions.

With conditions (2.19)-(2.24), it is possible to uniquely determine the coefficients $a_i$, $b_i$, $c_i$ and $d_i$. Let $h_i = x_{i+1} - x_i$, for $i = 0, \ldots, n-1$. Remembering that $s_i(x) = a_i(x-x_i)^3 + b_i(x-x_i)^2 + c_i(x-x_i) + d_i$, one obtains the following values for the derivatives:

$$s_i'(x_{i+1}) = 3a_ih_i^2 + 2b_ih_i + c_i, \tag{2.25}$$

$$s_i''(x_{i+1}) = 6a_ih_i + 2b_i. \tag{2.26}$$

From condition (2.19), $d_i = y_i$ is clear. Introducing new variables, let

$$\sigma_i = s''(x_i) = \begin{cases} s_i''(x_i) & = 2b_i, \text{ for } i = 0, \ldots, n-1, \\ s_{n-1}''(x_n) & = 6a_{n-1}h_{n-1} + 2b_{n-1}, \text{ for } i = n. \end{cases} \tag{2.27}$$

From this definition,

$$b_i = \frac{\sigma_i}{2}, \text{ for } i = 0, \ldots, n-1. \tag{2.28}$$

From definition (2.27), condition (2.23) and equation (2.26),

$$\sigma_{i+1} = s_{i+1}''(x_{i+1}) = s_i''(x_{i+1}) = 6a_ih_i + 2b_i, \text{ for } i = 0, \ldots, n-2,$$

which, combined with the second case of definition 2.27, gives

$$a_i = \frac{\sigma_{i+1} - \sigma_i}{6h_i}, \text{ for } i = 0, \ldots, n-1. \tag{2.29}$$

From conditions (2.19), (2.20) and (2.21), and using $d_i = y_i$,

$$y_{i+1} = a_ih_i^3 + b_ih_i^2 + c_ih_i + y_i, \text{ for } i = 0, \ldots, n-1. \tag{2.30}$$

Inserting (2.28) and (2.29) into (2.30) gives

$$y_{i+1} = \frac{\sigma_{i+1} - \sigma_i}{6h_i}h_i^3 + \frac{\sigma_i}{2}h_i^2 + c_ih_i + y_i, \text{ for } i = 0, \ldots, n-1,$$

which is rearranged to get

$$c_i = \frac{y_{i+1} - y_i}{h_i} - h_i \frac{2\sigma_i + \sigma_{i+1}}{6}, \text{ for } i = 0, \ldots, n-1. \tag{2.31}$$

Condition (2.22) and equation (2.25) relate $c_i$ and $c_{i+1}$ by

$$c_{i+1} = s'_{i+1}(x_{i+1}) = s'_i(x_{i+1}) = 3a_i h_i^2 + 2b_i h_i + c_i, \text{ for } i = 0, \ldots, n-2. \tag{2.32}$$

Inserting (2.28), (2.29) and (2.31) into (2.32) gives

$$\frac{y_{i+2} - y_{i+1}}{h_{i+1}} - h_{i+1}\frac{2\sigma_{i+1} + \sigma_{i+2}}{6} = 3\frac{\sigma_{i+1} - \sigma_i}{6h_i}h_i^2 + \sigma_i h_i + \frac{y_{i+1} - y_i}{h_i} - h_i\frac{2\sigma_i + \sigma_{i+1}}{6},$$

which simplifies to

$$h_i\sigma_i + 2h_i\sigma_{i+1} + 2h_{i+1}\sigma_{i+1} + h_{i+1}\sigma_{i+2} = 6\left(\frac{y_{i+2} - y_{i+1}}{h_{i+1}} - \frac{y_{i+1} - y_i}{h_i}\right). \tag{2.33}$$

Equation (2.33) is valid for $i = 0, \ldots, n-2$, yielding $n-1$ equations.

Using the not-a-knot conditions (2.24),

$$\begin{aligned} s'''_0(x_1) = 6a_0 = 6a_1 = s'''_1(x_1) &\Rightarrow a_0 = a_1 \\ (2.29) &\Rightarrow \frac{\sigma_1 - \sigma_0}{6h_0} = \frac{\sigma_2 - \sigma_1}{6h_1} \\ &\Rightarrow \sigma_0 = \left(1 + \frac{h_0}{h_1}\right)\sigma_1 - \frac{h_0}{h_1}\sigma_2. \end{aligned}$$

A similar reasoning yields $\sigma_n = \left(1 + \frac{h_{n-1}}{h_{n-2}}\right)\sigma_{n-1} - \frac{h_{n-1}}{h_{n-2}}\sigma_{n-2}$. Thus one can write the following tridiagonal system of $n-1$ equations:

$$\begin{bmatrix} \frac{h_0^2}{h_1} + 3h_0 + 2h_1 & -\frac{h_0^2}{h_1} + h_1 & 0 & \ldots & 0 \\ h_1 & 2h_1 + 2h_2 & h_2 & \ldots & 0 \\ 0 & h_2 & 2h_2 + 2h_3 & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & h_{n-3} \\ 0 & 0 & \ldots & h_{n-2} - \frac{h_{n-1}^2}{h_{n-2}} & 2h_{n-2} + 3h_{n-1} + \frac{h_{n-1}^2}{h_{n-2}} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \ldots \\ \sigma_{n-1} \end{bmatrix} = 6 \begin{bmatrix} \frac{y_2 - y_1}{h_1} - \frac{y_1 - y_0}{h_0} \\ \frac{y_3 - y_2}{h_2} - \frac{y_2 - y_1}{h_1} \\ \ldots \\ \ldots \\ \frac{y_n - y_{n-1}}{h_{n-1}} - \frac{y_{n-1} - y_{n-2}}{h_{n-2}} \end{bmatrix}$$

Solving this system gives the values of $\sigma_i$ for $i = 1, \ldots, n-1$, thus also giving the value for $\sigma_0$ . With these $\sigma_i$, equations (2.29), (2.28) and (2.31) are used to find the values of $a_i$, $b_i$ and $c_i$ for $i = 0, \ldots, n-1$, respectively, so all $4n$ coefficients are identified.

The spline $s(x)$ can be used as an approximation for a function $f$ which passes through the points $(x_0, y_0)$,...,$(x_n, y_n)$. This approximation is fourth order accurate, so the error is bounded as follows [1]:

$$\max_{a \le x \le b} \mid f(x) - s(x) \mid \le c \max_{a \le x \le b} \mid f''''(x) \mid \max_{0 \le i \le n-1} h_i^4$$

for some constant $c$.

For the spline approximation of the trans-membrane potential $u$, the output of *ode45* $(t_j, u(t_j, \tau))$ are used as knots. Having set a maximal step-size for *ode45*, which is usually quite small, the term $\max\limits_{0 \le i \le n-1} h_i^4$ is very small so the approximation is very accurate.

# Chapter 3

# Parameter identification

In this chapter, the parameter identification problem is presented. Section 3.1 exhibits the optimization problems that are to be solved. Section 3.2 presents in detail four different optimization methods that are used to solve the optimization problems. A proof of the convergence of the four methods is found in section 3.3.

## 3.1 Optimization Problems

### 3.1.1 General problem

The parameters found in mathematical models are variable and have effects on the properties of the solutions. Therefore, the problem of identifying which set of parameters yields desired properties, like matching experimental data for example, can be formulated as

$$\text{Find } \tau^* \text{ that minimizes } J, \tag{3.1}$$

where $\tau \in \mathbb{R}^n$ is the vector containing the values of the parameters in a chosen model and $J = J(\tau)$ is the function that represents how well the solution $y(\tau)$ of the model with parameters $\tau$ satisfy the target properties. Problem (3.1) is a minimization problem. What distinguishes one problem from another is how $J$, which is called the cost function, is defined.

### 3.1.2 Optimization problem I

Based on what is covered in chapter 2, the MS model can be used to predict the durations of each phase of the AP. Experimental data is available for these durations, so the

following $J$ can be considered:

$$J(\tau) = \sum_{i=1}^{4} \omega_i (P_i - P_i^*)^2, \tag{3.2}$$

where $P_i$ is the duration of phase $i$ in the solution $y(\tau) = [u(t,\tau), v(t,\tau)]$ of (2.1)-(2.3) with parameters $\tau = [\tau_{in}, \tau_{out}, \tau_{open}, \tau_{close}] \in \mathbb{R}^4$ and $I_{stim}$ restricted to only one stimulation. The $P_i$ are computed as in section 2.3.2. $P_i^*$ is the target duration of phase $i$, often given by experimental data, and $\omega_i$ is a weight assigned to scale the error on the different phase durations. It is convenient to write $P^* = [P_1^*, P_2^*, P_3^*, P_4^*]$. Since the phases have significantly different durations, a difference of 0.5 ms between $P_1$ and $P_1^*$ should be more impactful than that same difference between $P_4$ and $P_4^*$, for example. Setting $\omega_i = \frac{1}{(P_i^*)^2}$ makes the differences $(P_i - P_i^*)^2$ relative to $P_i^*$. This choice makes accuracy for shorter phases more important than for longer ones.

An important observation on the problem associated with (3.2) (abbreviated as problem (3.2) for the remainder of the thesis) is that there are no explicit constraints on $\tau$. However, not all $\tau$ actually induce an AP for which phase durations can be calculated (e.g. oddly shaped or non-existent AP). This points towards the need for constraints on the values of $\tau$, but such bounds are unknown. A slight modification to (3.2) takes care of this ambiguity:

$$J(\tau) = \begin{cases} \sum_{i=1}^{4} \frac{(P_i - P_i^*)^2}{(P_i^*)^2} & \text{if the } P_i \text{ are successfully computed} \\ \infty & \text{otherwise.} \end{cases} \tag{3.3}$$

The alternate value $\infty$ for $J$ is chosen because the cost function is to be minimized, so $\tau$ for which no phase durations are found are not minimizer candidates.

Optimization problem I is chosen for a few reasons. Firstly, problem I was used and shown to be efficient in [24], so a comparison can be made with the results in that work for testing and calibrating of the optimization methods presented in section 3.2. Secondly, work done in [29] shows that there is essentially a one to one relationship between the phase durations and the parameters $\tau$, so it is a natural choice to use the phase durations as a way of identifying $\tau$.

### 3.1.3 Optimization problem II

The MS model can be used to match multiple consecutive APs. Consider a transmembrane potential $\tilde{u}_i = \tilde{u}_i(t)$, $t \in [0, T_i]$, for some $i \in \mathbb{N}$, measured experimentally and

normalized in a certain way. Consider also a scaling factor $s_i > 0$. Define $J_i = J_i(\tau, s_i)$ by

$$J_i(\tau, s_i) = \frac{\int_0^{T_i} \mid u_i(t,\tau) - s_i\tilde{u}_i(t) \mid^2 dt}{\int_0^{T_i} \mid s_i\tilde{u}_i(t) \mid^2 dt}, \tag{3.4}$$

where $y_i(\tau) = [u_i(t,\tau), v_i(t,\tau)]$ is the solution of (2.1)-(2.3) with parameters $\tau$ and $I_{stim,i}$ is adjusted to match the stimulation pattern of $\tilde{u}_i$. The normalization and use of scaling factors are discussed further in section 5.1. Consider multiple different $\tilde{u}_i$, $i = 1, \ldots, N$, obtained by stimulating the same heart at different frequencies and find $[\tau^*, S^*]$ minimizing

$$J(\tau, S) = \sum_{i=1}^{N} J_i(\tau, s_i), \tag{3.5}$$

where $S = (s_1, \ldots, s_N)$ are the scaling factors for the $\tilde{u}_i$.

This problem differs from (3.3) and is more challenging to solve. Firstly, instead of only considering one AP, problem (3.5) tries to match data that includes multiple consecutive APs so that restitution properties may be accounted for. Since $I_{stim}$ is adjusted to stimulate the cardiac cell more than once, each AP beyond the first does not start from equilibrium (2.3). This affects the shape of the AP. Secondly, this problem takes into account the entire shape of the AP given by the model, not just the time between certain threshold values. Thirdly, this problem introduces $N$ new variables into $J$ in the form of the scaling factors $S$, increasing the number of variables from 4 to $4+N$.

The choice of optimization problem II is also inspired by work done in [24], where a similar problem is used to fit a single AP of recorded data. Problem II on the other hand accounts for multiple APs across multiple frequencies in order to better capture restitution properties in the identification of the parameters $\tau$. The formulation of the cost function (3.4)-(3.5) resembles what is done in [32] and is the usual least-square function used to fit a curve to another. The inclusion of the denominator in (3.4) scales each term in (3.5), which is needed because the datasets $\tilde{u}_i$ have different lengths $T_i$ over which a varying number of APs is observed.

## 3.2 Optimization methods

Many methods already exist to optimize functions. Usually, a standard route to explore is derivative-based methods such as the gradient method (steepest descent). However, work done in [24] shows that this method does not converge when applied to this parameter

identification problem. To summarize, continuity and differentiability of $J$ in $\tau$ could not be proven, which is not unreasonable as $J$ depends on equation (2.2) which contains a discontinuity. Also, $J$ is a composite function, so any of the inner functions might be causing problems. For example, $J$ in (3.3) is of the form

$$\tau \mapsto [u(t), v(t)] \mapsto P = [P_1, P_2, P_3, P_4] \mapsto J(\tau),$$

where the dependence on $\tau$ is implicit through $u$, $v$ and $P$. Attempts to remedy the discontinuity by using a continuous right-hand-side in (2.2) for the model still did not work. This remark hints towards making use of non-differentiable optimization strategies. One such strategy and three of its variations of our own design are considered.

The idea behind the methods explored in this work is that even though the gradient of an objective function can be hard or costly to compute, the fact that it exists can be useful. Even in the case where the gradient does not exist everywhere, the following notions could still be useful to analyze the convergence of the proposed methods. A few definitions are in order.

**Definition 3.2.1.** *Given a real function $f$ of $x \in \mathbb{R}^n$, a vector $d$ is called a descent direction for $f$ at $x$ if there exists $\alpha^{max} > 0$ such that for all $\alpha \in \mathbb{R}$ with $0 < \alpha < \alpha^{max}$, the following inequality holds:*

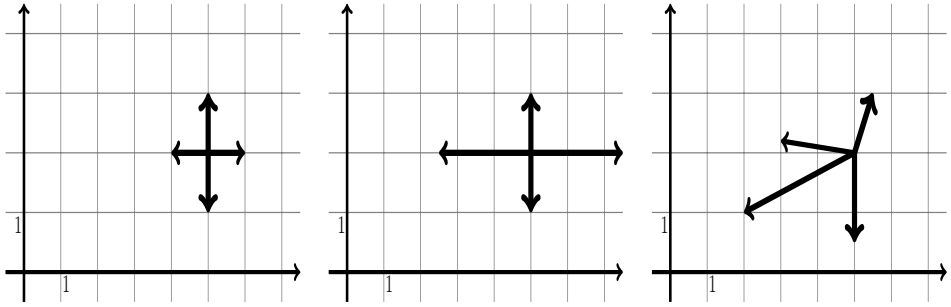
$$f(x + \alpha d) < f(x).$$

Simply, a descent direction is a direction for which small enough steps along this one always yield decrease in $f$. Finding a descent direction for $f$ at a point $x$ instead of using $-\nabla f(x)$ is another way of decreasing the value of the function. The following property can be used:

**Lemma 3.2.2.** *For a function $f$ and a point $x \in \mathbb{R}^n$ at which $f$ is differentiable, the following are equivalent:*

$$-\nabla f(x)^T d > 0 \Leftrightarrow d \text{ is a descent direction for the function } f \text{ at } x.$$

This says that if $d$ is within 90° of the negative of the gradient of a function at a point, it is a descent direction for that function at that point. In this spirit, to find a descent direction, one can consider a set of directions for which any vector is within 90° of at least one of those directions. This leads to the following definition (taken from [13]):




(a) Orthonormal set (b) Orthogonal set (c) Non-orthogonal set

Figure 3.1: Three different generating sets in $\mathbb{R}^2$, centered at the point $x = (5, 2)$

**Definition 3.2.3.** *A set $D = \{d^{(1)}, d^{(2)}, \ldots, d^{(p)}\}$ of $p$ vectors in $\mathbb{R}^n$ is called a generating set of $\mathbb{R}^n$ if for any $v \in \mathbb{R}^n$, there exist $c_i \geq 0$ such that*

$$v = \sum_{i=1}^{p} c_i d^{(i)}.$$

In other words, a generating set is like a spanning set with the added restriction that the coefficients for the linear combination must be positive. Figure 3.1 shows three examples of generating sets in $\mathbb{R}^2$. In particular, the following result [4] illustrates why generating sets are important in this context:

**Lemma 3.2.4.** *$D$ is a generating set of $\mathbb{R}^n$ if and only if for any $v \in \mathbb{R}^n$, there exists $d \in D$ such that*

$$v^T d > 0.$$

If a generating set is found, it is hence guaranteed that one of its elements is a descent direction, since it will be within 90° of $-\nabla f(x)$. The way of choosing this set is the main ingredient of the optimization strategies described below.

Finally, in order to avoid ambiguity on the meaning of the word "direction", it is understood throughout the rest of the thesis that "direction" and vector are synonyms. So, unlike the standard definition of the word "direction" in linear algebra, in this context a direction has an orientation.

### 3.2.1 Compass Search

**Algorithm 3.2.5** (Compass Search-CS). *Given a function $f$, an initial guess $x \in \mathbb{R}^n$, an initial direction-size $\delta > 0$, a contraction factor $c \in (0,1)$ and a stopping criterion stop, the following is applied:*

***while*** *stop is not met*

*let* $D = \{p\delta e_i \mid p = -1, +1$ *and* $e_i$ *is an element of the canonical basis for* $\mathbb{R}^n\}$

*let* $x^* = x + \arg\min_{d \in D} f(x+d)$

***if*** $f(x^*) < f(x)$, ***then***

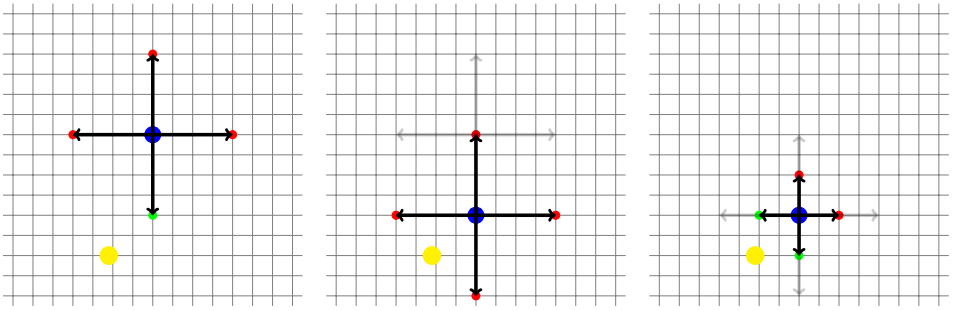
*set* $x \leftarrow x^*$

***else***

*set* $\delta \leftarrow c\delta$

Taken from [13], this method is the simplest and the foundation for the three other methods introduced here. Given a point $x \in \mathbb{R}^n$, a "canonical" generating set is considered. This canonical generating set is obtained by using $\pm e_i$, where $e_i$ are the elements of the canonical basis for $\mathbb{R}^n$. Denoting the length of the directions by $\delta$, the canonical generating set $D$ is constructed. Once again, it is understood that a direction is simply a vector in this context and so the direction-size is simply the length of the vector. Figure 3.1 (a) shows an example of $D$ in $\mathbb{R}^2$. $f$ is then evaluated at the points resulting from taking a step of size $\alpha = 1$ in a direction $d \in D$. The point that yields the smallest value of $f$, noted $x^*$, is considered. If $f(x^*) < f(x)$, then $x$ is moved to the new point $x^*$. Otherwise, no step of size 1 for all $d \in D$ made the function decrease. However, being a generating set, $D$ contains at least one descent direction for $f$ at $x$ if $\nabla f(x) \neq 0$. So the only problem is that the steps are too large (i.e. $1 > \alpha^{max}$ in definition 3.2.1). The direction-size $\delta$ used in creating $D$ is reduced by a factor of $c$, making the directions $d \in D$ shorter for the next iteration.

Figure 3.2 shows examples of three consecutive iterations of Compass Search in $\mathbb{R}^2$. The initial guess $x$ or its modified position is the blue dot. The set $D$ of four directions $d \in D$ is illustrated by the black arrows. The $D$ for the previous iteration is illustrated by the shaded arrows. The red dots are the points $x + d$ for which the function is greater than $f(x)$ and the green ones are those where the function is smaller. The yellow dot is the minimizer CS is looking for. In subfigure (a), the set $D$ yields a green dot, so $x$ is moved there for the next iteration. In subfigure (b), no direction results in a decrease of $f$, so the direction-size is contracted for the next iteration. In subfigure (c), two points yield decrease so the algorithm will choose the one with the smallest value of $f$.



(a) initial $D$ (b) next iteration (c) contraction

Figure 3.2: An example of the first 3 iterations of Compass Search in $\mathbb{R}^2$.

The algorithm repeats the process of constructing $D$ and moving $x$, if possible, as long as the stopping criterion is not met. Options for the stopping criterion include $f(x) \leq ftol$ for a chosen tolerance $ftol$ on the value of $f$, $\delta \leq \delta tol$ for a chosen tolerance $\delta tol$ on the direction-size, the number of iterations $k \geq k_{max}$ and the number of function evaluations $fevals \geq fevals_{max}$ among many others. The different parameters that appear in CS, such as the contraction factor $c$ and the stopping criterion *stop*, can be modified to make the algorithm better adapted to a specific minimization problem.

In the context of minimizing problem (3.3), certain details about Compass Search need clarification or modification. Here, $\tau \in \mathbb{R}^4$, so the generating set $D$ will contain 8 directions. However, $\tau = [\tau_{in}, \tau_{out}, \tau_{open}, \tau_{close}]$ has components with different scales ($\tau_{in} \ll \tau_{out} \ll \tau_{open},\ \tau_{close}$). Hence, a direction-size $\delta$ that is comparable to $\tau_{in}$ is too small for $\tau_{open}$, for example. Conversely, a $\delta$ comparable to $\tau_{close}$ is too large for $\tau_{out}$. Simply taking the same $\delta$ for each coordinate can lead to either insignificant changes in one direction or exaggerated changes in another, hence affecting the performance of the algorithm. For this reason, $\delta$ is taken to be relative to the coordinate it is changing. $D$ then looks like

$$D = \{p\delta\tau_i e_i \mid p = -1, +1 \text{ and } e_i \text{ is an element of the standard basis for } \mathbb{R}^4\}, \quad (3.6)$$

where $\tau_i$ is understood to be one on the four coordinates of $\tau$ (typically, $[\tau_1, \tau_2, \tau_3, \tau_4] = [\tau_{in}, \tau_{out}, \tau_{open}, \tau_{close}]$). The direction-size $\delta$ now represents the percentage by which each coordinate is changed instead of the absolute size of the change itself. The choice $0 < \delta < 1$ is appropriate, since $\tau_i > 0$ is required in this context. If $\delta \geq 1$, the four directions that decrease $\tau_i$ would yield points that have $\tau_i \leq 0$, which is not permitted.

Figure 3.1 (b) shows how this modification changes the way $D$ looks. In this figure,

$x = (5, 2)$ and $\delta = 0.5$. Hence, $D$ contains two longer directions and two shorter ones, since the coordinates of $x$ have different values.

The contraction factor $c$ is originally chosen to be $\frac{1}{2}$ in [13], but in this work, some effort was put in to see the effect of varying $c$. The conclusion of this analysis is that choosing $c$ closer to 1 offers more reliable results than choosing $c$ closer to 0, such as better minimizers or faster computation times. Details on this are found in section 4.3. For all tests, the value for $c$ is specified when results are given.

For problem (3.3), the stopping criterion is chosen so that it is met when either the total number of function evaluations exceeds a given maximum or the value of the function is smaller than a given tolerance. Since the function $J$ to be minimized involves solving the model (2.1)-(2.3) numerically, each function evaluation takes considerably more time than other tasks in the algorithm. Thus, the number of function evaluations is a good estimate for the total computational time and the stopping criterion limits this number. Also, the goal of parameter identification is to find parameters $\tau$ for which the solution of the model has phase durations equal to some target values. The stopping criterion reflects when this goal is reached. When each phase has the desired duration, then $J = 0$. Since it is not reasonable to expect the algorithm to find the exact $\tau^*$ for which $J(\tau^*) = 0$, either because it simply does not exist or because finding the exact right direction-size to land on this $\tau^*$ is unlikely, the minimizer is considered to be found if $J(\tau^*) \leq ftol$ with $ftol$ very small.

For problem (3.5), the same comments as above apply, subject to changing a few sentences so that they makes sense in the new context (e.g. the goal is no longer to match phase durations, but to fit the whole curve).

### 3.2.2 Golden Compass Search

**Algorithm 3.2.6** (Golden Compass Search-GCS). *Given a function $f$, an initial guess $x \in \mathbb{R}^n$, an initial direction-size $\delta > 0$, a contraction factor $c \in (0,1)$, a stopping criterion stop and a one-dimensional, non-differentiable optimization method 1-DOM, the following is applied:*

***while*** *stop is not met*

*let* $D = \{p\delta e_i \mid p = -1, +1$ *and* $e_i$ *is an element of the canonical basis for* $\mathbb{R}^n\}$

*let* $x^* = x + \arg\min_{d \in D} f(x+d)$

***if*** $f(x^*) < f(x)$, ***then***

*let* $d^* = \arg\min_{d \in D} f(x+d)$

*Consider* $\phi(\alpha) = f(x + \alpha d^*)$

*Apply 1-DOM to* $\phi$ *on* $[0,1]$ *to find* $\alpha^* = \arg\min_{[0,1]} \phi(\alpha)$

***if*** $f(x + \alpha^* d^*) < f(x^*)$, ***then***

*set* $x \leftarrow x + \alpha^* d^*$

***else***

*set* $x \leftarrow x^*$

***else***

*set* $\delta \leftarrow c\delta$

The Compass Search algorithm gives a way of finding a descent direction, but it makes no attempt at dealing with the step-size $\alpha$ (see definition 3.2.1). It simply takes $\alpha = 1$ once a descent direction is found. Choosing the right value for $\alpha$ is a way to greatly increase the efficiency of each step taken. The Golden Compass Search is a variant of CS that attempts to pick the optimal step-size $\alpha^*$ or at least a better one.

The initial step for constructing the set $D$ is the same as in CS. As before, whether or not $D$ contains a direction that creates a point where $f$ is smaller is determined, but the difference (highlighted in red in the GCS algorithm) lies in the case where $D$ contains a direction that yields decrease in $f$. The direction $d^*$ that leads to the largest decrease in $f$ is considered and the function $\phi(\alpha)$ is minimized for smaller steps along $d^*$. An optimization method, called 1-DOM, is used on $\phi(\alpha)$ to find an optimal step-size $\alpha^* \in [0,1]$. Any one-dimensional, non-differentiable optimization method 1-DOM can be used to do the one-dimensional minimization, but the one chosen for GCS is the Golden-Section Search, which explains the name Golden Compass Search.

**Algorithm 3.2.7** (Golden-Section Search-GSS)**.** *Given a function* $\phi : [a, b] \to \mathbb{R}$*, an interval* $[a, b]$ *and a stopping criterion stop, the following is applied*

***while*** *stop is not met*

$\quad$ *let* $c = a + \frac{b-a}{\rho^2}$

$\quad$ *let* $d = a + \frac{b-a}{\rho}$

$\quad$ ***if*** $\phi(c) < \phi(d)$, ***then***

$\quad\quad$ *set* $b \leftarrow d$

$\quad$ ***else***

$\quad\quad$ *set* $a \leftarrow c$

Note that $\rho = \frac{1+\sqrt{5}}{2}$ is the golden ratio (hence the name of the algorithm). Golden-Section Search is a well known method for minimizing a function over an interval [3]. Given an initial interval, GSS shrinks the interval at each iteration progressively enclosing the minimizer within closer lower and upper bounds. When the interval length is sufficiently small or the maximum number of iterations is reached (this is the usual stopping criterion *stop*), any value in the interval is assumed to be a good approximation for the minimizer. The standard choice is $\alpha^* = \frac{a+b}{2}$.

Given that GSS finds an approximation of the minimizer and that $\phi$ may lack unimodality [3], the point $x+\alpha^* d^*$ might not be the actual minimizer (there are pathological cases that can arise). As a precaution, both the points $x + \alpha^* d^*$ and $x^*$ are considered. $x$ is moved to whichever one has a smaller value for $f$ and then the process can repeat itself as in CS.

Coming back to the problems (3.3) and (3.5), the same comments made about Compass Search apply to Golden Compass Search. The rescaled way of constructing $D$ given in (3.6), the choice of stopping criterion and the choice of contraction factor are all the same as before. The goal of GCS is to make every step better at reducing the function, so in terms of iterations of the method, it should do better than CS. This means that $D$ will be generated less often and there will be less function evaluations in that regard. However, each of these better steps is now more cpu-intensive, since the function must be re-evaluated many times during the GSS. For this work, the stopping criterion for GSS limits the number of iterations to 3. Whether this is an efficient trade-off in terms of total function evaluations is seen with the results in section 4.4.

### 3.2.3 Hybrid Compass Search

**Algorithm 3.2.8** (Hybrid Compass Search-HCS). *Given a function $f$, an initial guess $x \in \mathbb{R}^n$, an initial direction-size $\delta > 0$, a contraction factor $c \in (0,1)$ and a stopping criterion stop, the following is applied:*

***while*** *stop is not met*

*let* $D = \{p\delta e_i \mid p = -1, +1$ *and* $e_i$ *is an element of the canonical basis for* $\mathbb{R}^n\}$

*let* $x^* = x + \arg\min_{d \in D} f(x+d)$

***if*** $f(x^*) < f(x)$, ***then***

*set* $\Delta f = 0$

*let* $D' = \{d \in D \mid f(x+d) < f(x)\}$

***for*** $d \in D'$

*set* $\Delta f \leftarrow \Delta f + (f(x) - f(x+d))$

***for*** $d \in D'$

*let* $d_d^* = \frac{f(x)-f(x+d)}{\Delta f} d$

*let* $x^{**} = x + \sum_{d \in D'} d_d^*$

***if*** $f(x^{**}) < f(x^*)$, ***then***

*set* $x \leftarrow x^{**}$

***else***

*set* $x \leftarrow x^*$

***else***

*set* $\delta \leftarrow c\delta$

The principle behind the computation of a generating set is to find a set which is guaranteed to contain a descent direction, but a generating set can be arbitrarily large. Once a generating set is found, adding any vectors to it does not change the fact that it still contains at least one descent direction. Hybrid Compass Search focuses on improving the generating set so that more efficient directions are considered.

Once again, the algorithm starts as for Compass Search. The idea is to take the subset $D'$ of $D$ that contains the directions along which a step of size 1 reduces the function and take a "hybrid" of them to create a new, hopefully better direction along which a larger decrease of $f$ is observed. With this in mind, it is advantageous to try to make a smart choice so that more often than not, the hybrid is the best option and the algorithm did not "waste time" by adding a new direction in $D$. For this work, the hybrid is created by taking an element of the cone generated by the directions in $D'$. The hybrid is a weighted average of all the directions in $D'$. The weight that each $d \in D'$ has

is the decrement of $f$ by taking that direction divided by the sum $\Delta f$ of the decrements of $f$ for all $d \in D'$. The weights hence lie in $[0, 1]$ and sum up to 1. The larger decrease in $f$ a direction leads to, the more that direction contributes to the weighted sum to build the hybrid. In the special case where there is only one direction in $D'$, the hybrid is the same as that direction, since its weight is 1. In this case, the algorithm could skip the step of creating the hybrid to save time, but the principle remains.

There are pathological cases where the hybrid chosen is not actually a better choice than the initial directions in $D$. For example, if the current $x$ is a saddle point and $D'$ contains a direction that increases coordinate and a direction that decreases that same coordinate, their hybrid could end up being $\vec{0}$. To account for this uncertainty, the hybrid is added to the directions in $D$ and the direction yielding the largest decrease is chosen (the hybrid is not chosen if it is not the best option).

Figure 3.3 shows an example of how this method creates a hybrid direction in $\mathbb{R}^2$. Since both the left and down directions yield decrease (green dots), the hybrid (light gray) of those two directions is also considered. The light gray diamond shape is a guideline for all the possible hybrids. In this case, it is supposed for example that going down makes the function decrease twice as much as going left, so the hybrid is 2/3 of a step down and 1/3 of a step left.

The parameters included in Hybrid Compass Search ($\delta$, $c$, the rescaled $D$, etc.) take the same values as for Compass Search when applied to both problems (3.3) and (3.5). Compared to CS, HCS focuses on making a better guess at the optimal descent direction, so each iteration is more effective and the set $D$ needs to be generated fewer times. However, each iteration now costs one extra function evaluation (to check if the hybrid direction is better). The results in section 4.4 show if this is a good compromise or not.

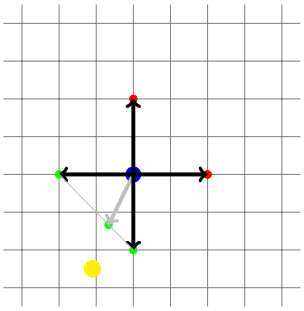


Figure 3.3: An example of the hybrid creation step in Hybrid Compass Search

### 3.2.4 Front-track Compass Search

**Algorithm 3.2.9** (Front-track Compass Search-FCS). *Given a function $f$, an initial guess $x \in \mathbb{R}^n$, an initial direction-size $\delta > 0$, a contraction factor $c \in (0,1)$, an expansion factor $w > 1$ and a stopping criterion stop, the following is applied:*

***while*** *stop is not met*

  *let* $D = \{p\delta e_i \mid p = -1, +1$ *and* $e_i$ *is an element of the canonical basis for* $\mathbb{R}^n\}$

  *let* $x^* = x + \arg\min_{d \in D} f(x+d)$

  ***if*** $f(x^*) < f(x)$, ***then***

    *let* $d^* = \arg\min_{d \in D} f(x+d)$

    *set* $x^{**} = x^*$

    ***while*** $f(x^{**}) \leq f(x^*)$

      *set* $x^* \leftarrow x^{**}$

      *set* $d^* \leftarrow wd^*$

      *set* $x^{**} \leftarrow x^{**} + d^*$

    *set* $x \leftarrow x^*$

  ***else***

    *set* $\delta \leftarrow c\delta$

An issue that the three previous methods may have arises when checking $D$ for directions that yield decrease. The points $x+d$ for $d \in D$ are used to determine whether $d$ is potentially a descent direction. However, as mentioned before, it is possible that step of size $\alpha = 1$ may be too large and yield increase in the value of $f$, even though $d$ could be a descent direction. This may lead to $\delta$ becoming too small too quickly and since

the three other methods do not consider taking step of size larger than 1, the algorithm might make very minimal progress because the directions are too small. There is hence a need to separate the length of the directions used to check for decrease and the length of the step taken to create a new point $x^* = x + \alpha d$.

There are ways of ensuring that an algorithm does not take steps that are too small, such as imposing a sufficient decrease criterion like the Armijo condition [19] or using backtracking. Backtracking consists of taking a larger step than necessary and then progressively shortening the step until an acceptable step length is found. Acceptable could mean satisfying the Armijo condition, for example, or it could also mean that the chosen step length is just large enough so that a smaller step would not decrease the function as much. Front-track Compass Search tries to remedy the problem of small steps in a different way.

Front-track Compass Search is somewhat similar to Golden Compass Search in that it does a sort of one-dimensional minimization, but it goes about it differently. It allows choosing $\alpha^* > 1$. Once the direction $d^*$ is found, the same way it is in CS, FCS makes the largest progress possible along $d^*$ by taking increasingly larger steps in that direction until taking another step would make $f$ start increasing. A point of the form $x^* = x + \alpha^* d^*$ with $\alpha^* \geq 1$ is found as the minimizer along the direction $d^*$. The algorithm gets its name from the fact that instead of the standard method of backtracking, FCS takes progressively larger steps and "front-tracks" to the minimizer, as illustrated in figure 3.4. Technically, all three of the previous methods employ a backtracking method in the form of the last line of the **while** loop (set $\delta \leftarrow c\delta$). Note that this is the same last line for FCS, so it too can be considered a backtracking method, but it has the added novelty of doing front-tracking first.

The expansion factor $w$ is chosen larger than 1 so that the front-tracking steps get larger and hopefully the process ends sooner, but not so much larger than 1 that only a few steps can be taken before the process ends and the minimizer is not as well approximated. An example of front-tracking with a large $w$ is shown in figure 3.5.

The rescaled way of constructing $D$ in (3.6) is also used in Front-track Compass Search for problems (3.3) and (3.5), but other modifications are included. The innermost **while** loop of FCS consists of repeatedly taking larger steps using the chosen direction $d^*$. With a rescaled $D$, $d^*$ is now relative to the current $x$. This remains true during the innermost **while** loop. Each additional step rescales $d^*$ to the current $x^{**}$. The FCS algorithm with

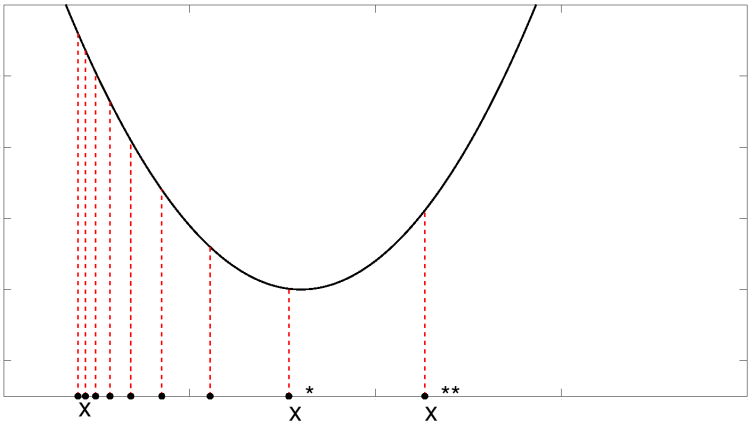



Figure 3.4: An example of work done by FCS during the inner **while** loop. The black dots are the $x$ obtained for each step, with the last two points labeled $x^*$ and $x^{**}$ to illustrate the moment when the function starts growing (i.e. $f(x^*) < f(x^{**})$)

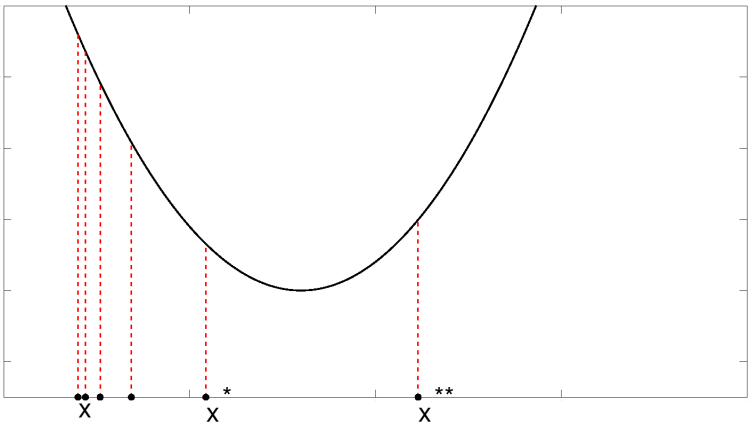



Figure 3.5: An example of the word done by FCS during an inner **while** loop where the expansion factor $w$ is large. The black dots are the $x$ obtained for each step, with the last two points labeled $x^*$ and $x^{**}$ to illustrate the moment the function starts growing (i.e. $f(x^*) < f(x^{**})$)

the rescaled $D$ proceeds as follows, with changes highlighted in blue:

**Algorithm 3.2.10** (Front-track Compass Search with rescaled $D$-FCS)**.** *Given a function $f$, an initial guess $x \in \mathbb{R}^n$, an initial direction-size $\delta > 0$, a contraction factor $c \in (0,1)$, an expansion factor $w > 1$ and a stopping criterion stop, the following is applied:*

***while*** *stop is not met*

  *let* $D = \{p\delta x_i e_i \mid p = -1, +1$ *and* $e_i$ *is an element of the canonical basis for* $\mathbb{R}^n\}$

  *let* $x^* = x + \arg\min_{d \in D} f(x+d)$

  ***if*** $f(x^*) < f(x)$, ***then***

    *let* $d^* = \arg\min_{d \in D} f(x+d)$

    *set* $x^{**} = x^*$

    *set* $\delta^* = \delta$

    ***while*** $f(x^{**}) \leq f(x^*)$

      *set* $x^* \leftarrow x^{**}$

      *set* $\delta^* \leftarrow w\delta^*$

      *set* $x^{**} \leftarrow x^{**} + p\delta^* x_i^{**} e_i$ *for the appropriate* $p$ *and* $i$

    *set* $x \leftarrow x^*$

  ***else***

    *set* $\delta \leftarrow c\delta$

There are also notable differences between FCS and the other three variants of CS when it comes to applying them to solve problems (3.3) and (3.5). The stopping criterion remains unchanged, but the values of $\delta$ and $c$ are quite different. Keeping in mind the comment about the length of the directions used to check for decrease made earlier, $\delta$ is chosen to be smaller than with other CS methods. While this smaller $\delta$ leads to $D$ covering less of the parameter space with FCS, taking larger step with front-tracking should compensate. Also, $c$ is much smaller than in CS so that FCS has more space to work with. FCS takes a small initial step to find the potential descent direction and then takes multiple steps along it. If the new direction length $c\delta$ is too close to $\delta$, there would be too little space for FCS to take multiple steps. The increased number of step-sizes tested at each iteration will increase the number of function evaluations, but more progress is made at each iteration so $D$ should be generated fewer times, thus decreasing the number of function evaluations. Again, the results in section 4.4 illustrate the efficiency of FCS in comparison with other CS methods.

## 3.3 Proof of convergence

The goal of the four methods presented is to find a minimizer of a function and the results in chapters 4 and 5 show that they are reliable in doing so, at least for the parameter identification problems (3.3) and (3.5). However, it is desirable for these methods to be guaranteed to work if a given set of conditions is satisfied.

The terms "global convergence" and "local convergence" in the context of this work, following [13] describe the ability that an optimization method has to converge to a stationary point of a function (i.e. the ability to find point where the gradient vanishes). A method is said to converge globally if the values of $x$ at each iteration of the method, denoted by $x^k$, converge to a stationary point regardless of the initial guess. In symbols, $\lim_{k\to\infty} x^k = x^*$ with $\nabla f(x^*) = 0$. A method converges locally if the $x^k$ converge to a stationary point when the initial guess is "close enough".

The article [13] from which Compass Search is taken provides a lengthy discussion on global and local convergence of what is referred to as generating set search methods. Generating set search methods represent a larger class of methods that have a similar structure. CS, as well as the three other variations presented in section 3.2, are examples of generating set search methods. The implementation of these methods differs a bit from the implementation described in the article, mostly due to the modification done to $D$ in equation (3.6). Because of this different implementation, some of the results on convergence require slightly modified proofs while some proofs simply do not apply. What follows is a rewriting of some proofs and some explanations on why other proofs do not work. For more details and a comparison of the approaches, one can read this section in conjunction with chapters 2 and 3 of [13].

Firstly, some notations are required, most of which are taken from [13]. As eluded to in the description of the methods, an iteration consists in building $D$ for the point $x$, checking to see if decrease is found using directions in $D$, possibly doing some extra steps to make a better choice for the next point and finally either moving $x$ or contracting $\delta$. An iteration is said to be successful if $x$ is moved to a point with a smaller function value and it is said to be unsuccessful if $\delta$ is contracted because no better point was found. If the $k$th iteration is successful, $k \in \mathbb{S}$ and if the $k$th iteration is unsuccessful, $k \in \mathbb{U}$. $\mathbb{S}$ and $\mathbb{U}$ are thus disjoint, possibly infinite subsets of $\mathbb{N}$.

The values of $x$ and $\delta$ at the beginning of iteration $k$ are denoted $x^k$ and $\delta^k$. $x^0$ is the initial guess and $\delta^0$ is the initial direction-size.

Let $\mathcal{D} = \{\pm e_i \mid e_i \text{ is an element of the canonical basis for } \mathbb{R}^n\}$. For many of the

proofs in [13] as well as lemma 3.3.1 and theorem 3.3.2, it is required that the generating set remain fixed. To account for the fact that $D$ is chosen as in (3.6), let $\Delta_i^k = \delta^k x_i^k$. With this notation, $D = \{z \mid z = \Delta_i^k d,\ d = \pm e_i \in \mathcal{D},\ i = 1, 2, \ldots, n\}$. Also, the points resulting from a step of size $\alpha = 1$ along directions in $D$ can all be written as $x^k + \Delta_i^k d$ for the corresponding $d$. This differs from [13], where $\Delta^k$ was the same for all $d \in \mathcal{D}$.

The differences between the four methods presented arise during successful iterations in the way the next iterate $x^{k+1}$ is chosen, but for unsuccessful iteration, they are all the same. Fortunately, the behaviour at unsuccessful iterations is the key for convergence.

**Lemma 3.3.1.** *Let $f$ be a continuously differentiable, real-valued function on $\mathbb{R}^n$ with $\nabla f$ Lipschitz with constant $M$. Then for unsuccessful iterations of CS, GCS, HCS and FCS,*

$$\|\nabla f(x^k)\| \leq \sqrt{n} M \Delta_i^k, \text{ for } k \in \mathbb{U} \text{ and for some } i = i(k) \in \{1, \ldots, n\}. \tag{3.7}$$

*Proof.* Let $k \in \mathbb{U}$. Then

$$f(x^k) \leq f(x^k + \Delta_i^k d), \text{ for all } d \in \mathcal{D}. \tag{3.8}$$

Lemma 6.3 in [33] states that, for any value of $\nabla f(x)$, there exists at least one $d \in \mathcal{D}$, say $d = \pm e_i$, $i = i(k)$, such that

$$\frac{1}{\sqrt{n}} \|\nabla f(x^k)\| \, \|d\| \leq -\nabla f(x^k)^T d. \tag{3.9}$$

From (3.8) and the mean value theorem, there exists an $\alpha^k \in [0, 1]$ such that

$$0 \leq f(x^k + \Delta_i^k d) - f(x^k) = \nabla f(x^k + \alpha^k \Delta_i^k d)^T \Delta_i^k d.$$

Subtracting $\Delta_i^k \nabla f(x^k)^T d$ from both sides gives

$$-\Delta_i^k \nabla f(x^k)^T d \leq \Delta_i^k (\nabla f(x^k + \alpha^k \Delta_i^k d) - \nabla f(x^k))^T d.$$

Dividing both sides by $\Delta_i^k$ and applying (3.9) gives

$$\frac{1}{\sqrt{n}} \|\nabla f(x^k)\| \, \|d\| \leq (\nabla f(x^k + \alpha^k \Delta_i^k d) - \nabla f(x^k))^T d. \tag{3.10}$$

Using the fact that $\nabla f$ is Lipschitz with constant $M$ and taking the norm of the right-hand side of (3.10),

$$\|(\nabla f(x^k + \alpha^k \Delta_i^k d) - \nabla f(x^k))^T d\| \leq M \|x^k + \alpha^k \Delta_i^k d - x^k\| \, \|d\| \leq M \Delta_i^k \|d\|^2. \tag{3.11}$$

Combining (3.11) with (3.10) and using the fact that $\|d\| = 1$,

$$\|\nabla f(x^k)\| \leq \sqrt{n} M \Delta_i^k.$$

□

Lemma 3.3.1 gives a bound on the norm of the gradient at unsuccessful iterations that is proportional to $\Delta_i^k$. If it can be shown that $\lim_{k\to\infty} \Delta_i^k = 0,\ k \in \mathbb{U}$, then the conclusion is that the gradient also goes to zero and so the method converges to a stationary point.

Three different ways of showing that $\lim_{k\to\infty} \Delta_i^k = 0,\ k \in \mathbb{U}$, are presented in [13]. Thus, it is shown that, under certain conditions, the generating set search methods converge globally. But upon closer inspection, the three ways of ensuring global convergence (sufficient decrease, rational lattice and moving grids [13]) do not apply to CS and the other methods as implemented in this thesis. Presented here are a few comments on why the globalization techniques do not carry over. These comments are not meant to be specific, just general guidelines as to where the proofs in [13] fail for the CS methods.

Firstly, the sufficient decrease globalization in section 3.7.1 of [13] does not apply because it requires a sufficient decrease criterion to be considered when updating $x$ whereas the CS methods only use a simple decrease criterion.

Secondly, the rational lattice globalization in section 3.7.2 of [13] requires that the set of all directions considered throughout the algorithm is finite and that the length of the steps to check for decrease in the value of $f$ is of the form $\|d\| = \Lambda^\Gamma \delta^0$, for a fixed rational $\Lambda$ and an integer $\Gamma$. By considering the set of directions to be fixed to $\mathcal{D}$, which is finite, the first requirement is fulfilled, but the second requirement fails. The length of the steps is of the form $\|d\| = c^\Gamma \delta^0 x_i^k = \Delta_i^k$ in this case. Because of this difference, $x^{k+1} - x_0$ is no longer a positive integral linear combination of the search directions. It is a now a rational linear combination which renders an argument about finiteness of a certain set invalid and the proof no longer applies.

Thirdly, the moving grids globalization in section 3.7.3 of [13] requires that $D$ only be updated after unsuccessful iterations to ensure that a lattice (grid) structure is maintained, which is not the case here. Considering $\mathcal{D}$ as the fixed set of directions still causes similar problems as in the rational lattice case because the $\Delta_i^k$ change the grid at each iteration, not only the unsuccessful ones.

Even though the global convergence result can not be proved with these three globalization techniques, another proof may eventually be obtained for the CS methods. However, a more modest result is proved here instead. The following theorem guarantees

the local convergence of CS, GCS, HCS and FCS under certain conditions. This theorem is analogous to theorem 3.15 of [13].

**Theorem 3.3.2.** *Let $f$ be a twice differentiable, real-valued function of $\mathbb{R}^n$. Suppose that $x^*$ is a local minimizer of $f$ and that $\nabla^2 f(x^*)$ is positive definite. For the CS methods, if $x^0$ is sufficiently close to $x^*$, $\delta^0$ is sufficiently small and $\lim_{k\to\infty} \delta^k = 0$, then it is guaranteed that*

$$\lim_{k\to\infty} x^k = x^*.$$

*Proof.* Let $\eta > 0$ be such that

$$x \in \mathcal{B}(x^*, \eta) \Rightarrow spectrum \nabla^2 f(x) \subset [\frac{\phi_{min}}{2}, 2\phi_{max}], \tag{3.12}$$

where $\phi_{min}$ and $\phi_{max}$ are the minimum and maximum eigenvalues of $\nabla^2 f(x^*)$, respectively. Let $\xi = \max_{x\in\mathcal{B}(x^*,\eta)} \{ \mid x_i \mid \mid i = 1, \ldots, n\}$. From (3.12), as in the proof of theorem 3.15 in [13], the following inequality is obtained

$$\frac{\phi_{min}}{2}\|x^k - x^*\| \leq \|\nabla f(x^k)\|.$$

Using lemma 3.3.1, for $k \in \mathbb{U}$ and some $i = i(k) \in \{1, \ldots, n\}$,

$$\|x^k - x^*\| \leq \frac{2}{\phi_{min}}\sqrt{n} M \Delta_i^k \leq \frac{2}{\phi_{min}}\sqrt{n} M \xi \delta^k. \tag{3.13}$$

Next, exactly as in the proof of theorem 3.15 in [13], it can be shown that there exist $c$ such that

$$\|x - x^*\| \leq c \, \|y - x^*\| \text{ for any } x, y \in \mathcal{B}(x^*, \eta) \text{ with } f(x) \leq f(y). \tag{3.14}$$

Assume that $x^0 \in \mathcal{B}(x^*, \eta)$ is close enough to $x^*$ so that

$$\|x^0 - x^*\| \leq \frac{\eta}{2c}.$$

For CS, GCS and HCS (FCS is dealt with later), assume $\delta^0$ small enough so that

$$\mid \delta^0 \mid \leq \frac{\eta}{2\xi}.$$

With these assumptions, if $x^k \in \mathcal{B}(x^*, \eta)$, then $x^{k+1} \in \mathcal{B}(x^*, \eta)$ as well. Indeed,

$$\|x^{k+1} - x^*\| \leq \|x^{k+1} - x^k\| + \|x^k - x^*\|. \tag{3.15}$$

From the updating formulae

$$\begin{aligned} x^{k+1} &= x^k + \Delta_i^k d \text{ for CS,} \\ x^{k+1} &= x^k + \alpha^* \Delta_i^k d,\ 0 \le \alpha^* \le 1 \text{ for GCS and} \\ x^{k+1} &= x^k + z,\ z \in D' \cup \{hybrid\} \text{ for HCS,} \end{aligned}$$

an upper bound for the first term in (3.15) is found in all cases:

$$\|x^{k+1} - x^k\| \le \Delta_i^k \le \delta^0 \xi \le \frac{\eta}{2} \tag{3.16}$$

and since $f(x^k) \le f(x^0)$ and $x^0, x^k \in \mathcal{B}(x^*, \eta)$, (3.14) is applied to get an upper bound for the second term in (3.15):

$$\|x^k - x^*\| \le c\, \|x^0 - x^*\| \le \frac{\eta}{2}. \tag{3.17}$$

Thus, (3.15),(3.16) and (3.17) combine to say that $x^{k+1} \in \mathcal{B}(x^*, \eta)$ for all $k = 0, 1, 2, \ldots$. Let $k \in \mathbb{S}$ and $\omega(k)$ denote the most recent unsuccessful iteration. Then by (3.13) and (3.14),

$$\|x^k - x^*\| \le c\, \|x^{\omega(k)} - x^*\| \le c \frac{2}{\phi_{min}} \sqrt{n} M \xi \delta^{\omega(k)}. \tag{3.18}$$

(3.13) and (3.18) together with the assumption $\lim_{k\to\infty} \delta^k = 0$ give $x^k \to x^*$.

For FCS, it is necessary to enforce a maximal number $q$ of steps during each inner **while** loop. Assume $\delta^0$ small enough so that

$$\mid \delta^0 \mid \le \frac{\eta}{2qw^q\xi}.$$

where $w$ is the expansion factor used in FCS. The updating formula, which is dependent on the number of front-tracking steps made, generally looks something like

$$x_i^{k+1} = x_i^k (1 \pm w^0 \delta^k)(1 \pm w^1 \delta^k) \ldots (1 \pm w^r \delta^k), \text{ for some integer } r < q.$$

The idea is that if a maximum of $q$ front-tracking steps are taken and each of those steps gets progressively bigger, than the entire change on $x^k$ is bounded above by $q$ times the biggest possible step, i.e.

$$\|x^{k+1} - x^k\| \le q w^q \delta^0 \xi \le \frac{\eta}{2}.$$

Having found an upper bound analogous to (3.16), the proof continues as before. One should notice that the assumption that $\delta^0$ be small enough is more restrictive for FCS than for CS, GCS and HCS. The upper bound on $\delta^0$ is smaller for FCS. This corroborates the choice of a smaller $\delta^0$ for FCS, as mentioned in section 3.2.4. □

# Chapter 4

# Results for problem I

In order to test the efficiency of the optimization methods presented in section 3.2, the optimization problem (3.3) is used. In section 4.1, the methods are first calibrated by studying the effect of the parameters used in *ode45* on the cost function (3.3). Section 4.2 contains an in-depth look at a typical numerical test. In section 4.3, the effect of the contraction factor $c$ on the performance of the methods is investigated. Finally, section 4.4 presents numerical results for a selection of numerical tests to compare the methods with each other.

A few remarks are in order before showing numerical results. As previously mentioned, for problem (3.3), $I_{stim}$ is restricted to stimulate the cell only once. The parameter $A$ in equation (2.9) was calculated with $v^* = 0.7$ and $\beta = 0.05$. The choice $v^* = 0.7$ instead of $v^* = 1$ was made at the time to try to adjust $I_{stim}$ to work for both a single and multiple stimulations. By analyzing equation (2.9), it is seen that setting $v^*$ to be smaller than it actually should be leads to an increase in $A$, which can affect the duration of phase I or even overstimulate. However, since the tests are done only for verification purposes, the former concern is overlooked while the latter did not seem to occur. The stimulation is applied for $\Delta t = 2$ ms.

## 4.1 *ode45* parameters

Upon conducting preliminary tests to compare the performance of the four methods presented in section 3.2, the accuracy of the minimizers was questioned. The parameters $\tau$ and the phase durations $P$ recovered from the optimization methods were not as close to $\tau^*$ and $P^*$ as expected. Some effort was put in to find ways of improving the accuracy

of the minimizer. In particular, adjusting the tolerances was considered. The function tolerance was initially set to $ftol = 10^{-4}$ in preliminary tests. It was sometimes possible to reach this tolerance using the methods while other times the tolerance was not reached. Suspecting that this was a problem with the cost function rather than a problem with the optimization methods, the smoothness of the cost function was investigated.

The goal was to see how the function was behaving in the standard directions used by the algorithms. One property that was especially looked at was the presence of "noise" or "jaggedness" that would cause a multitude of local minima to be present, eventually resulting in the method to terminate at any local minimizer rather than the global minimizer.

Figure 4.1 shows the cost function for $P^* = [7.021, 251.091, 34.322, 270.350]$ obtained from $\tau^* = [0.3, 6, 130, 150]$ using the following *ode45* parameters:

$$MaxStep = 0.5,\ InitialStep = 0.01,\ RelTol = 10^{-4} \text{ and } AbsTol = 10^{-4}. \tag{4.1}$$

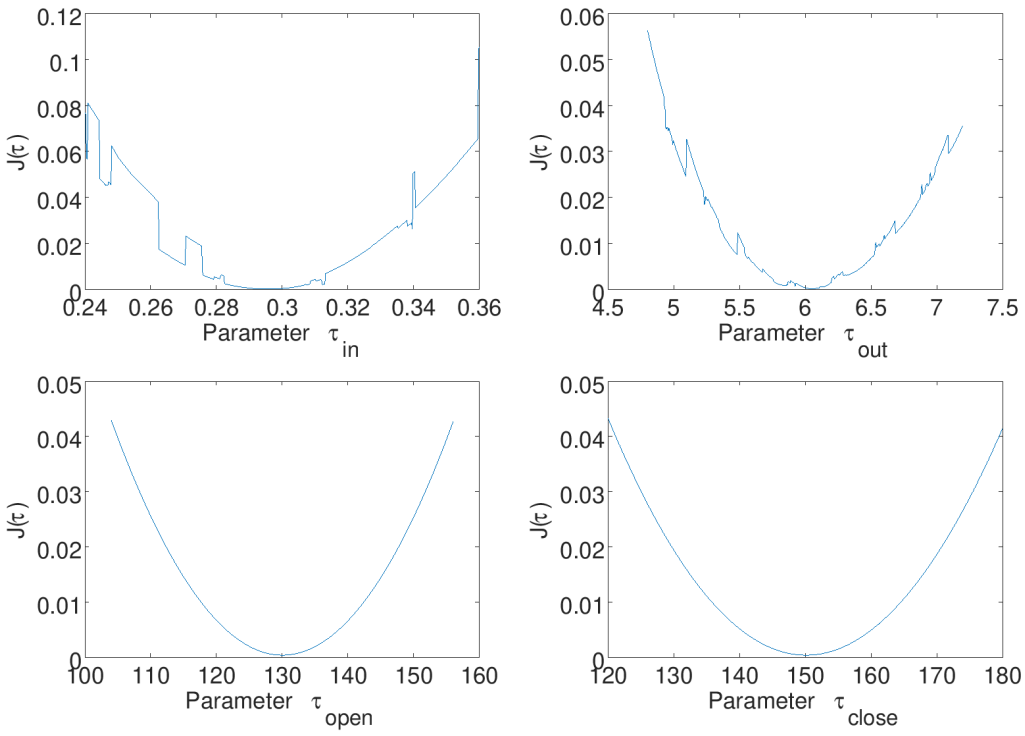



Figure 4.1: Graphs of $J(\tau)$ as a function of $\tau_{in}$ (top left), $\tau_{out}$ (top right), $\tau_{open}$ (bottom left) and $\tau_{close}$ (bottom right) using *ode45* parameters given in (4.1).

The cost function is indeed very noisy, at least in two directions (along the $\tau_{in}$-axis and the $\tau_{out}$-axis). Varying the ode45 parameters to smooth-out the cost function yielded figure 4.2, which shows the cost function for $P^* = [6.892, 251.283, 34.313, 270.427]$ obtained from $\tau^* = [0.3, 6, 130, 150]$ using the following *ode45* parameters:

$$MaxStep = 0.5,\ InitialStep = 0.01,\ RelTol = 10^{-12} \text{ and } AbsTol = 10^{-12}. \tag{4.2}$$

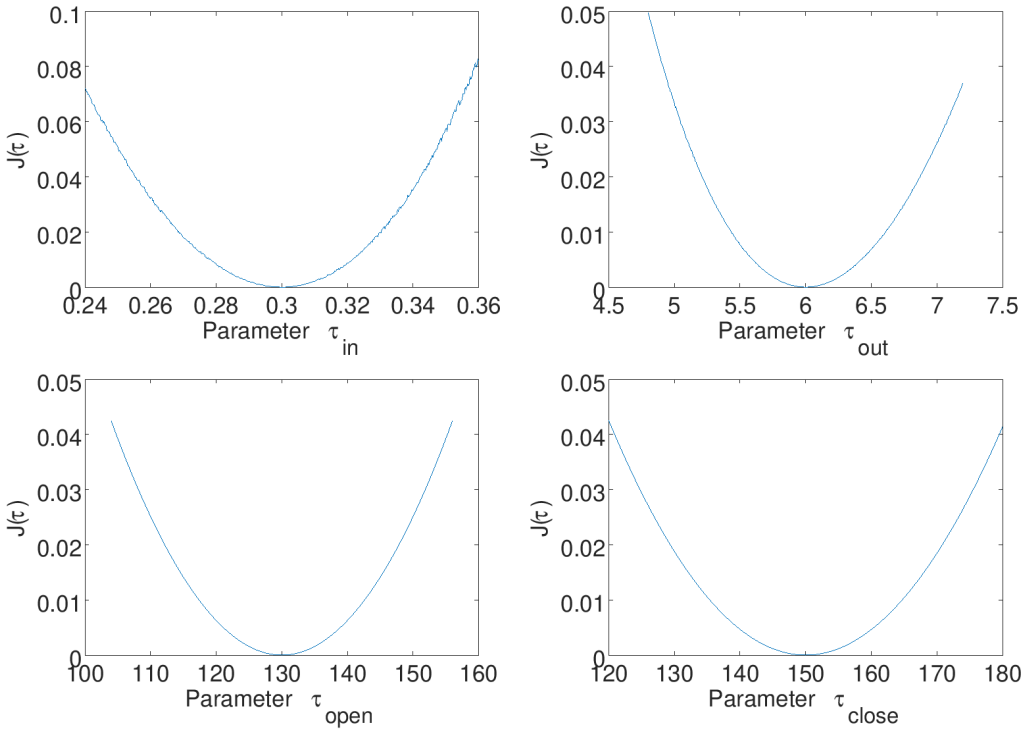



Figure 4.2: Graphs of $J(\tau)$ as a function of $\tau_{in}$ (top left), $\tau_{out}$ (top right), $\tau_{open}$ (bottom left) and $\tau_{close}$ (bottom right) using *ode45* parameters given in (4.2).

For this set of *ode45* parameters, the cost function is smoother. Of course, zooming in reveals the same noisy behaviour as before, but the noise occurs on a smaller scale. Setting *RelTol* and *AbsTol* to smaller values give more accurate numerical solutions of the ODE and values of the cost function, but at the expense of an increased computing time. Smaller tolerances could have been used, but the results obtained using values from (4.2) were deemed satisfactory for soling the parameter identification problem. Hence, for other results presented in further sections, unless specified otherwise, *ode45* parameters given in (4.2) are used.

## 4.2 Sample numerical test case

Presented here in detail is a typical numerical test case for solving problem (3.3). A similar careful analysis is done for all test cases in the thesis.

One of the tests presented in section 4.4, whose results appear in table 4.7, is considered. This test involves using Compass Search to identify parameters that give phase durations matching $P^* = [8.685, 251.840, 48.734, 205.919]$ obtained from parameters $\tau^* = [0.4, 10, 100, 130]$. The initial guess is $\tau = [0.3, 6, 130, 150]$. The stopping criterion is $f \leq 10^{-8}$ or $fevals \geq 1500$.

In addition to outputting the final values of the parameters $\tau_{final}$, the values of the phase durations $P_{final}$, the value of the cost function $J(\tau_{final})$ and the total amount of function evaluations $fevals$, CS also outputs information at each iteration. A sample of the output of CS for this test is found in figure 4.3.

| Iteration | Action | CostFunction | FEvals |
|---|---|---|---|
| 0 | Initial | 0.22832220128602 | 1 |
| 1 | Decrease parameter 3 | 0.13091466094901 | 9 |
| 2 | Increase parameter 1 | 0.10210503463520 | 17 |
| 3 | Increase parameter 2 | 0.04135031344045 | 25 |

Figure 4.3: A sample of the output of Compass Search for the sample test case

To illustrate the convergence of the method, a convergence graph (see figure 4.4) is created by plotting the value of the cost function on a logarithmic scale with respect to the number of function evaluations. Also featured on this graph are red dots marking the points where certain tolerances on the value of $J$ are reached to illustrate where the algorithm would terminate if the function tolerance criterion was less demanding. The actual function tolerance criterion used for this test is $f \leq 10^{-8}$ (indicated by a dashed line on the graph).

A characteristic feature of the optimization methods used in this thesis is the monotonic decrease of the value of the cost function with iterations. This is due to fact that the algorithms only change the value of the variable, in this case $\tau$, if doing so reduces the cost function. This is called a simple decrease criterion [13].

Another advantage of seeing the progress made at each iteration of the algorithm is being able to see when progress has stopped or is not significant. The term "stagnate" is introduced and used whenever the algorithm continues to adjust parameters and to

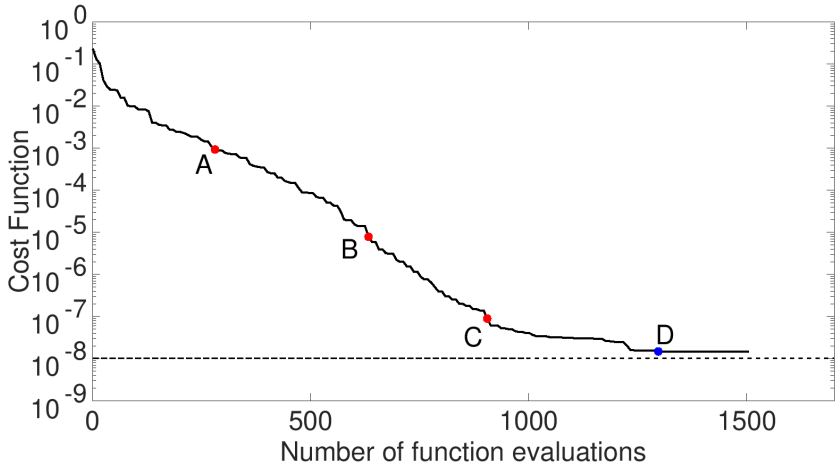



Figure 4.4: Value of the cost function plotted with respect to the number of function evaluations for the sample test case. The red dots mark the points where the cost function reaches a specified tolerance: (A) $10^{-3}$, (B) $10^{-5}$ and (C) $10^{-7}$. The blue dot (D) marks the point where the algorithm stagnates.

decrease $\delta$ while only making minimal progress on diminishing the cost function. An algorithm will be said to have stagnated at an iteration when all iterations beyond it do not affect the cost functions first three significant digits. On the convergence graph, stagnation occurs when the graph becomes almost horizontal until the maximal number of function evaluations is reached. Looking at figure 4.4 and the entire output of CS shows that, in the sample test, CS stagnates at iteration 162, after 1297 function evaluations.

To have a better understanding of the impact of the function tolerance criterion on the quality of the minimizer $\tau_{final}$, tables 4.1 and 4.2 show the statistics of this sample test for different tolerance values. Note that the points A, B and C in figure 4.4 correspond to the stopping points for the tolerances $10^{-3}$ (A), $10^{-5}$ (B) and $10^{-7}$ (C). For a function tolerance of $10^{-3}$, the $P_{i,final}$ found are accurate to around 1-2%. Lower function tolerances such as $10^{-5}$ lead to much better minimizers where $P_{i,final}$ are accurate to around 0.1-0.2%. A tolerance on the cost function of about $10^{-5}$-$10^{-6}$ is thus sufficient to get accurate values of $\tau$ and $P$ for most test cases.

Table 4.1: Values of $\tau_{final}$ and $P_{final}$ for various $ftol$

| tolerance | $fevals$ | $\tau_{final}$ | $P_{final}$ | $J(\tau_{final})$ |
|---|---|---|---|---|
| $10^{-3}$ | 281 | $[0.405, 10.11, 101.78, 127.39]$ | $[8.788, 246.636, 48.826, 209.804]$ | $9.26e{-4}$ |
| $10^{-5}$ | 633 | $[0.401, 9.97, 99.82, 130.33]$ | $[8.692, 251.664, 48.658, 205.474]$ | $7.79e{-6}$ |
| $10^{-7}$ | 905 | $[0.401, 10.00, 99.99, 130.16]$ | $[8.686, 251.807, 48.739, 205.873]$ | $8.88e{-8}$ |
| $10^{-8}$ | 1500 | $[0.401, 10.00, 100.01, 130.17]$ | $[8.686, 251.831, 48.739, 205.920]$ | $1.47e{-8}$ |

Table 4.2: Errors on $t^*$ and $P^*$ for various $ftol$

| tolerance | Difference $\mid \tau^* - \tau_{final} \mid$ | Difference $\mid P^* - P_{final} \mid$ | Difference $\mid (P^* - P_{final})/P^* \mid$ (%) |
|---|---|---|---|
| $10^{-3}$ | $[0.005, 0.11, 1.78, 2.61]$ | $[0.103, 5.204, 0.092, 3.885]$ | $[1.19, 2.07, 0.19, 1.88]$ |
| $10^{-5}$ | $[0.001, 0.03, 0.18, 0.33]$ | $[0.007, 0.176, 0.076, 0.445]$ | $[0.08, 0.07, 0.16, 0.22]$ |
| $10^{-7}$ | $[0.001, 0.00, 0.01, 0.16]$ | $[0.001, 0.033, 0.005, 0.046]$ | $[0.012, 0.013, 0.010, 0.022]$ |
| $10^{-8}$ | $[0.001, 0.00, 0.01, 0.17]$ | $[0.001, 0.009, 0.005, 0.001]$ | $[0.012, 0.004, 0.010, 0.000]$ |

## 4.3 Contraction factor

The optimal choice of the contracting factor $c$ is investigated. Preliminary tests were conducted using Compass Search to find $\tau^*$ that minimizes (3.3) for $P_i^*$ obtained from the solution of (2.1)-(2.3) for chosen parameters and for some experimental $P_i^*$. The initial direction-size is $\delta = 0.2$ and the stopping criterion is $f \leq 10^{-4}$ or $fevals \geq 1000$. The values of the parameters for *ode45* given in (4.1) are used. As discussed in section 4.1, these parameter values are not the best choice when evaluating the cost function. However, for the purpose of exploring the effect of $c$, it was judged unnecessary to demand a higher quality cost function. The idea is only to have a general sense of which values of $c$ give convergence of the CS method.

For test 1 (see table 4.3), the target durations $P^* = [12.409, 208.555, 41.848, 243.092]$ are obtained using parameters $\tau^* = [0.5, 8, 120, 145]$. CS uses the initial guess $\tau_0 = [0.7, 6.5, 108, 160]$. For test 2 (see table 4.4), the target durations $P^* = [12.125, 190.390, 36.323, 244.685]$ are obtained using parameters $\tau^* = [0.46, 6.6, 122, 145]$. CS uses the initial guess $\tau_0 = [0.3, 6, 140, 175]$. For test 3 (see table 4.5), the target durations are $P^* = [8, 380, 65, 320]$ taken from table 4.8, so $\tau^*$ is unknown. The initial guess is $\tau_0 = [0.35, 9, 155, 210]$.

By looking at these three tables in conjunction with the output of CS, a few conclusions were made. In general, using larger values for $c$ slows the algorithm down a

Table 4.3: Test 1

| $c$ | $fevals$ | $\tau_{final}$ | $P_{final}$ | $J(\tau_{final})$ |
|---|---|---|---|---|
| 0.95 | 1001 | $[0.507, 7.86, 120.08, 149.08]$ | $[12.336, 209.584, 41.605, 242.356]$ | $1.016e{-4}$ |
| 0.85 | 353 | $[0.508, 7.84, 121.12, 149.99]$ | $[12.398, 209.905, 41.597, 244.193]$ | $9.928e{-5}$ |
| 0.75 | 305 | $[0.509, 7.83, 120.98, 150.19]$ | $[12.438, 209.665, 41.568, 243.904]$ | $9.002e{-5}$ |
| 0.65 | 289 | $[0.507, 7.83, 120.82, 149.73]$ | $[12.395, 209.611, 41.553, 243.528]$ | $7.997e{-5}$ |
| 0.55 | 1001 | $[0.500, 7.55, 120.87, 152.89]$ | $[12.408, 210.122, 40.712, 243.029]$ | $7.931e{-4}$ |
| 0.45 | 1001 | $[0.501, 7.57, 120.90, 153.06]$ | $[12.409, 210.322, 40.783, 243.031]$ | $7.199e{-4}$ |

Table 4.4: Test 2

| $c$ | $fevals$ | $\tau_{final}$ | $P_{final}$ | $J(\tau_{final})$ |
|---|---|---|---|---|
| 0.95 | 1001 | $[0.463, 6.21, 123.01, 154.65]$ | $[12.152, 192.441, 35.383, 244.136]$ | $7.961e{-4}$ |
| 0.85 | 529 | $[0.462, 6.46, 122.45, 148.42]$ | $[12.142, 190.624, 35.977, 244.556]$ | $9.462e{-5}$ |
| 0.75 | 1001 | $[0.494, 6.19, 124.77, 163.32]$ | $[13.012, 190.792, 35.796, 244.665]$ | $5.569e{-3}$ |
| 0.65 | 1001 | $[0.495, 6.19, 124.79, 163.50]$ | $[13.012, 191.025, 35.820, 244.666]$ | $5.556e{-3}$ |
| 0.55 | 1001 | $[0.490, 6.14, 124.80, 164.11]$ | $[13.012, 191.643, 35.640, 244.659]$ | $5.749e{-3}$ |
| 0.45 | 1001 | $[0.497, 6.22, 124.79, 163.38]$ | $[13.012, 190.947, 35.933, 244.670]$ | $5.477e{-3}$ |

Table 4.5: Test 3

| $c$ | $fevals$ | $\tau_{final}$ | $P_{final}$ | $J(\tau_{final})$ |
|---|---|---|---|---|
| 0.95 | 1001 | $[0.377, 13.61, 152.48, 162.17]$ | $[8.035, 378.146, 65.694, 321.079]$ | $1.684e{-4}$ |
| 0.85 | 1001 | $[0.376, 13.60, 152.07, 162.48]$ | $[8.030, 379.099, 65.709, 320.207]$ | $1.395e{-4}$ |
| 0.75 | 289 | $[0.371, 13.56, 151.87, 162.02]$ | $[8.001, 379.611, 65.627, 319.710]$ | $9.634e{-5}$ |
| 0.65 | 1001 | $[0.387, 12.96, 152.05, 170.13]$ | $[7.969, 382.574, 63.893, 319.736]$ | $3.521e{-4}$ |
| 0.55 | 1001 | $[0.370, 13.16, 151.90, 164.93]$ | $[8.007, 381.351, 64.362, 319.823]$ | $1.102e{-4}$ |
| 0.45 | 113 | $[0.375, 13.24, 152.18, 164.94]$ | $[7.956, 380.578, 64.590, 320.442]$ | $7.493e{-5}$ |

bit. This is seen by comparing cases where the function tolerance criterion was met, since smaller $c$ values tended to make the algorithm reach the stopping criterion in fewer function evaluations. For example, $c$ values of 0.85, 0.75 and 0.65 each enabled CS to reach the tolerance in test 1, with CS using $c = 0.65$ requiring the smallest number of function evaluations.

The cases where the algorithm failed to reach the tolerance in the allotted maximal number of function evaluations also reveal a trend. A closer look at the progress made at each iteration revealed that once again, in most cases, smaller $c$ values tended to enable faster progress even though the algorithm would eventually stagnate short of the tolerance criterion. For test 1 in table 4.3, the algorithm did not stagnate for $c = 0.95$, as progress was still being made when the maximum number of evaluations was reached. The algorithm did stagnate for $c = 0.55$ and $c = 0.45$, stagnating after 329 evaluations and 345 evaluations respectively. For test 2 in table 4.4, the algorithm using $c$ values of 0.95, 0.75, 0.65, 0.55 and 0.45 stagnated after 937, 561, 337, 353 and 425 evaluations, respectively. The convergence graph of test 2 in figure 4.5 shows a comparison of the progress of CS for each value of $c$.

While smaller $c$ values enabled faster initial progress, larger $c$ values showed more reliability in reaching the function tolerance criterion, which is indicative of the ability of the algorithm to find the best minimizer possible. When using $c = 0.95$, CS did not stagnate in the first and third tests, perhaps only needing more evaluations to reach the goal, and stagnated close to the goal in test 2. In particular, $c = 0.85$ showed promising results, with CS reaching the tolerance in two test cases and falling just short in the third ($J(\tau_{final}) = 0.0001395$ instead of 0.0001).

Keeping in mind that a choice is made between speed and reliability, the value $c = 0.85$ is chosen as the default when testing Compass Search, Hybrid Compass Search and Golden Compass Search. Front-track Compass Search uses a different $c$ value for reasons already explained in section 3.2.4.

## 4.4 Comparison of the optimization methods

The four methods presented in section 3.2 were applied to problem (3.3). For all results presented in this section, the methods use the following values: For Compass Search, Golden Compass Search and Hybrid Compass Search, the initial direction-size is $\delta = 0.2$ and the contraction factor is $c = 0.85$ and for Front-track Compass Search, $\delta = 0.02$,

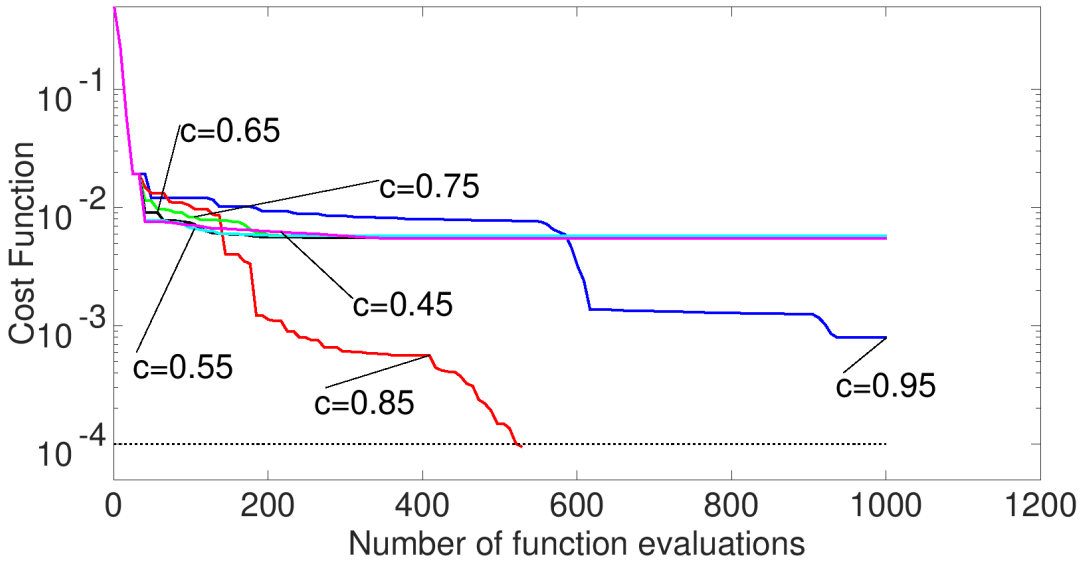



Figure 4.5: Value of the cost function plotted with respect to the number of function evaluations for test 2 with different $c$ values: 0.95 in blue, 0.85 in red, 0.75 in green, 0.65 in black, 0.55 in cyan and 0.45 in magenta.

$c = 0.25$ and $w = 1.25$. Different test cases, where $P^*$ were found using chosen $\tau^*$, were used to verify how well the methods perform. Since the global minimizer is known for these test cases, performance is more easily evaluated.

The first test case results are shown in table 4.6. Here, the target phase durations are $P^* = [6.892, 251.283, 34.313, 270.427]$, obtained from parameters $\tau^* = [0.3, 6, 130, 150]$. The initial guess is $\tau_0 = [0.27, 5.8, 127, 140]$. The stopping criterion is $f \leq 10^{-6}$ or $fevals \geq 1000$.

Table 4.6: Comparison of four methods: Test case 1

| Method | Stagnation | $fevals$ | $\tau_{final}$ | $P_{final}$ | $J(\tau_{final})$ |
|---|---|---|---|---|---|
| CS | None | 577 | $[0.2998, 6.003, 129.903, 149.931]$ | $[6.893, 251.357, 34.320, 270.248]$ | $5.747e{-7}$ |
| GCS | None | 493 | $[0.3003, 5.989, 130.030, 150.311]$ | $[6.895, 251.327, 34.284, 270.420]$ | $8.668e{-7}$ |
| HCS | None | 421 | $[0.3003, 5.995, 129.964, 150.179]$ | $[6.895, 251.581, 34.305, 270.303]$ | $4.043e{-7}$ |
| FCS | None | 148 | $[0.3003, 5.992, 130.025, 150.189]$ | $[6.897, 251.239, 34.591, 270.428]$ | $7.846e{-7}$ |

The second test case results are shown in table 4.7. Here, the target phase durations are $P^* = [8.685, 251.840, 48.734, 205.919]$, obtained from parameters $\tau^* = [0.4, 10, 100, 130]$. The initial guess is $\tau_0 = [0.3, 6, 130, 150]$. The stopping criterion is $f \leq 10^{-8}$ or

$fevals \geq 1500$.

Table 4.7: Comparison of four methods: Test case 2

| Method | Stagnation | $fevals$ | $\tau_{final}$ | $P_{final}$ | $J(\tau_{final})$ |
|---|---|---|---|---|---|
| CS | After 1297 | 1505 | $[0.4009, 9.999, 100.012, 130.166]$ | $[8.686, 251.831, 48.739, 205.920]$ | $1.472e{-8}$ |
| GCS | None | 1116 | $[0.4012, 9.994, 100.016, 130.256]$ | $[8.686, 251.845, 48.730, 205.919]$ | $9.381e{-9}$ |
| HCS | None | 831 | $[0.4012, 9.997, 100.017, 130.215]$ | $[8.685, 251.824, 48.737, 205.923]$ | $8.339e{-9}$ |
| FCS | After 579 | 1505 | $[0.4013, 9.993, 100.017, 130.277]$ | $[8.685, 251.850, 48.729, 205.918]$ | $1.497e{-8}$ |

Test case 1 shows that each method was able to meet the function tolerance stopping criterion, so each method was successful in finding a suitable minimizer. Comparing the $\tau_{final}$ found by the algorithms to the actual $\tau^*$ confirms this, the maximal relative error on the $\tau_i^*$ for any $\tau_{final}$ being 0.21%. Also, the maximal relative error on the $P_i^*$ for any $P_{final}$ is 0.82%. Looking at the number of function evaluations needed reveals that FCS performed the best, followed by HCS, then GCS and finally CS.

Test case 2 had an initial guess that was not as good as the one in test case 1 and also demanded higher accuracy on $J(\tau)$, so it was expected that more function evaluations would be required. GCS and HCS were successful in meeting the tolerance while CS and FCS stagnated not too far from reaching the required tolerance. The maximal relative error on the $\tau_i^*$ for any $\tau_{final}$ is 0.33% while the maximal relative error on the $P_i^*$ for any $P_{final}$ is 0.012%. For comparison's sake, considering that CS and FCS did almost reach the tolerance, the same performance ranking can be observed where FCS performed best while CS was the worst (if the stagnation point is considered to be the stopping point). One could also argue that HCS did better than FCS in the sense that a better minimizer was found, so it is understood that for this test, FCS and HCS are very close in terms of performance.

The four methods were also used to identify parameters that would match the experimental phase durations of the AP for different cardiac cells (see table 4.8). In these cases, the global minimizer $\tau^*$ is not known.

Table 4.8: Experimental durations (ms)

| Tissues | $P_1^*$ | $P_2^*$ | $P_3^*$ | $P_4^*$ |
|---|---|---|---|---|
| Left ventricle (LV) | 8 | 250 | 30 | 260 |
| Purkinje fibers (PF) | 8 | 380 | 65 | 320 |
| Right atria (RA) | $4-5$ | 100 | 20 | 250 |

Table 4.9 shows the results of the four methods applied to matching $P^*$ for a LV cell. The initial guess is $\tau_0 = [0.3, 6, 130, 140]$ and the stopping criterion is $f \leq 10^{-6}$ or $fevals \geq 500$. CS and HCS both successfully found a minimizer satisfying the function tolerance, but HCS required the smallest number of function evaluations. GCS failed to meet the function tolerance before reaching the maximum number of evaluations, but did not show signs of stagnation. Perhaps a higher number of function evaluations would permit GCS to find a suitable minimizer. FCS made its significant progress in 356 evaluations, which would have been quicker than HCS if FCS would not have stagnated. Still, FCS stagnated roughly at one order of magnitude above the tolerance and looking at the resulting $P_{final}$ shows that it was still close to matching $P^*$ (the largest relative error for any of the $P_{i,final}$ is $\approx 0.28\%$). For all methods, the maximal relative error on the $P_i^*$ for any $P_{final}$ is 0.6%.

Table 4.9: Comparison of four methods: LV

| Method | Stagnation | $fevals$ | $\tau_{final}$ | $P_{final}$ | $J(\tau_{final})$ |
|---|---|---|---|---|---|
| CS | None | 481 | $[0.3118, 4.606, 129.529, 185.740]$ | $[7.998, 249.823, 30.001, 260.158]$ | $9.169e{-7}$ |
| GCS | None | 507 | $[0.3130, 4.656, 129.106, 184.455]$ | $[8.003, 249.500, 30.178, 259.566]$ | $4.198e{-5}$ |
| HCS | None | 405 | $[0.3118, 4.607, 129.544, 185.745]$ | $[8.003, 249.867, 30.005, 260.198]$ | $9.962e{-7}$ |
| FCS | After 356 | 503 | $[0.3106, 4.584, 129.472, 186.173]$ | $[7.998, 250.171, 29.917, 259.997]$ | $8.108e{-6}$ |

Table 4.10 shows the results of the four methods applied to matching $P^*$ for a PF cell. The initial guess is $\tau_0 = [0.35, 9, 155, 210]$ and the stopping criterion is $f \leq 10^{-6}$ or $fevals \geq 500$. HCS and FCS both succeeded in finding a suitable minimizer, but FCS was roughly twice as fast, needing 269 fewer function evaluations. CS and GCS failed, but, with no stagnation, it is possible a bit more function evaluations were required. CS performed a bit better than GCS, yielding a final cost function value that is roughly 3 times smaller. For all methods, the maximal relative error on the $P_i^*$ for any $P_{final}$ is 0.28%.

Table 4.10: Comparison of four methods: PF

| Method | Stagnation | $fevals$ | $\tau_{final}$ | $P_{final}$ | $J(\tau_{final})$ |
|---|---|---|---|---|---|
| CS | None | 505 | $[0.3819, 13.345, 152.295, 165.244]$ | $[7.999, 379.413, 64.941, 320.513]$ | $5.787e{-6}$ |
| GCS | None | 501 | $[0.3813, 13.306, 151.778, 166.100]$ | $[7.990, 381.039, 64.892, 319.360]$ | $1.592e{-5}$ |
| HCS | None | 501 | $[0.3814, 13.354, 151.992, 165.337]$ | $[7.996, 379.982, 65.003, 319.830]$ | $5.213e{-7}$ |
| FCS | None | 232 | $[0.3806, 13.346, 152.090, 165.350]$ | $[7.998, 380.230, 64.978, 320.072]$ | $6.173e{-7}$ |

Table 4.11 shows the results of the four methods applied to matching $P^*$ for a RA

cell. The initial guess is $\tau = [0.15, 5, 75, 120]$ and the stopping criterion is $f \leq 10^{-5}$ or $fevals \geq 1000$. Each method succeeded, but FCS was the fastest, followed by HCS, then GCS and finally CS. Compared to the LV and PF cases, the RA case tolerance criterion was less demanding and the maximal number of evaluations was larger. This is because $P^*$ has notably smaller values, so that an error of the same magnitude increases the value of $J(\tau)$ more than in the previous two cases. Nevertheless, looking at the $P_{final}$ values shows that the methods matched $P^*$ well (a maximal relative error of 0.25%). The increased amount of available function evaluations compared to the LV and PF tests was chosen to observe if CS and GCS could find suitable minimizers given enough function evaluations.

Table 4.11: Comparison of four methods: RA

| Method | Stagnation | $fevals$ | $\tau_{final}$ | $P_{final}$ | $J(\tau_{final})$ |
|---|---|---|---|---|---|
| CS | None | 593 | $[0.1822, 4.214, 117.001, 54.003]$ | $[4.001, 100.119, 19.988, 250.479]$ | $5.593e{-6}$ |
| GCS | None | 549 | $[0.1825, 4.239, 116.698, 53.757]$ | $[3.993, 99.935, 20.049, 249.890]$ | $9.281e{-6}$ |
| HCS | None | 340 | $[0.1820, 4.227, 116.571, 53.827]$ | $[3.999, 100.022, 20.016, 249.603]$ | $3.341e{-6}$ |
| FCS | None | 268 | $[0.1822, 4.237, 116.967, 53.710]$ | $[3.998, 99.901, 20.039, 250.499]$ | $9.049e{-6}$ |

Based on the results of the five test cases presented as well as numerous other tests, whose results are omitted to avoid redundancy, the following observations are made:

- Given enough functions evaluations, each of the four methods is capable of identifying a set a parameters yielding phase durations that match experimental phase durations $P^*$ within a prescribed tolerance ($< 1\%$ relative error on each $P_i^*$ is expected).

- FCS is most often the fastest of the four methods, requiring considerably less function evaluations to reach a certain function tolerance. However, FCS stagnates short of the goal in a few test cases where other methods do not.

- HCS is almost always successful in meeting the function tolerance criterion and very rarely stagnates. It is also most often the second fastest method after FCS.

- GCS does not perform consistently. It is sometimes successful where other methods are not while other times it performs worse than every other method. Compared to HCS and FCS, GCS often requires considerably more function evaluations to reach a certain tolerance.

- CS is more consistent than GCS, often being able to reach a function tolerance, but also requiring many function evaluations to do so. CS is reliable albeit slow.

These observations lead to this conclusion: in most cases, FCS is the best method, followed by HCS in second place and GCS and CS are considered to be very close in terms of performance. This ranking of the methods is considered when moving on to different problems such as problem (3.5). FCS is the method of choice to be used first when confronting a new problem with HCS being the backup method should FCS yield unsatisfactory results.

Finally, it is worth noting that, while inspired by CS and generating set searches as described in [13], the three variations GCS, HCS and FCS are an original contribution and represent in most cases a significant improvement over CS.

# Chapter 5

# Results for Problem II

The Front-track Compass Search optimization method is used to identify parameters for which the MS model matches experimentally recorded APs. Section 5.1 details how the data is recorded as well as how the data and the trans-membrane potential given by the model are reformatted to be used in the cost function (3.5). In section 5.2, the method is verified by using the MS model to generate target data in various numerical tests. In section 5.3, the experimental datasets acquired as explained in section 5.1 are matched individually while in section 5.4, they are matched simultaneously using either three or six different pacing frequencies at a time.

## 5.1 Data acquisition and preparation

Problem (3.5) requires experimental data showing the entire AP of a cardiac cell. For the results presented in this section, the experimental data used were recorded at the Sunnybrook Research Institute. The following description on how data is recorded is provided by Mihaela Pop, with more details available in [25]. AP waves are recorded using voltage-based optical fluorescence imaging. The fluorescence dye (di4-ANEPPS) and uncoupler to block contraction (2,3 BDM) are injected into the coronary circulation of a healthy explanted swine heart perfused by a Langendorff system. The optical dye is excited with green light (∼530nm) while the emitted epicardial signals is filtered (>610nm) and captured by a high-speed CCD camera (MICAM02, BrainVision Inc. Japan) at 256 frames/second (Figure 5.1). The field of view is 184×124 pixels (12×10cm), yielding an ∼0.7mm spatial resolution. Several different stimulation frequencies are used to study the restitution properties. The relative change in fluorescence signal intensity ($\Delta F/F$)

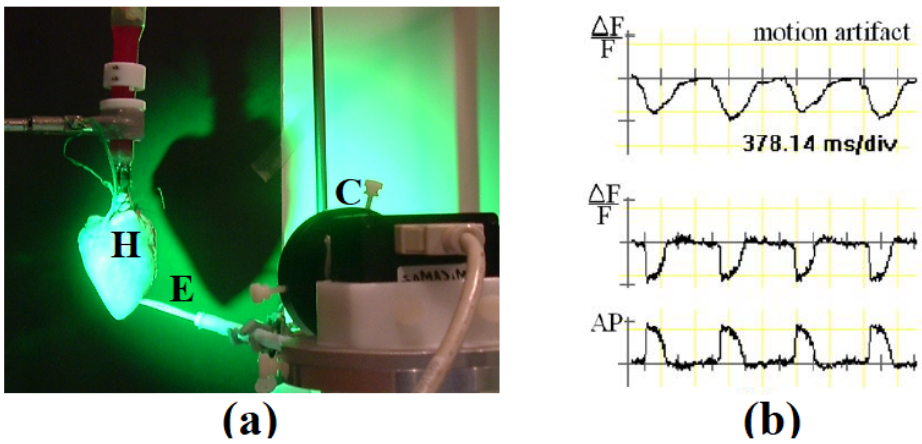



Figure 5.1: a) Snapshot of the optical experiment to record epicardial AP wave propagation using a fast CCD camera (C), where the pig heart (H) was stimulated via an electrode (E). (b) Examples of waves recorded at one pixel in the heart without the uncoupler (top) as well as after the uncoupler (bottom) was injected. Note that the inverse of the relative loss of fluorescence signal $\Delta F/F$ (arbitrary units) gives the AP. The waves were displayed with BV-Ana software (BrainVision, Japan).

recorded at each pixel, gives directly the AP waves. For model fitting, the AP waves recorded at one pixel selected from an area in the left ventricle (LV) where tissue was homogeneously illuminated, and also both fluorescence signal and tissue perfusion were homogeneous are used.

Once the AP waves are recorded, a set of points $(t_l, \frac{\Delta F_l}{F_l})$, $t_l \in [0, T]$, for each stimulating frequency is saved. These sets of points, denoted $(t_l, \hat{u}_i(t_l))$, are the raw experimental data that are used to define the $\tilde{u}_i(t)$ for the cost function (3.5). However, a normalization process must be applied to the data before matching the model.

Firstly, as mentioned in the caption of figure 5.1, the inverse of $\frac{\Delta F}{F}$ gives the AP, so the first operation done on the raw data is to "invert" it by doing

$$\hat{u}_i(t_l) \leftarrow \max_l(\hat{u}_i(t_l)) - \min_l(\hat{u}_i(t_l)) - \hat{u}_i(t_l).$$

Secondly, the model gives values $u_i(t) \in [0, 1]$, so the data must be normalized to be in that same range. Figure 5.2 shows that the raw data is not. In order to normalize the data, the "average" extrema must be calculated. The average values are considered, since taking the absolute minimum or maximum of a set of noisy values would not make sense when trying to normalize for this problem. The average minimum *amin* is found

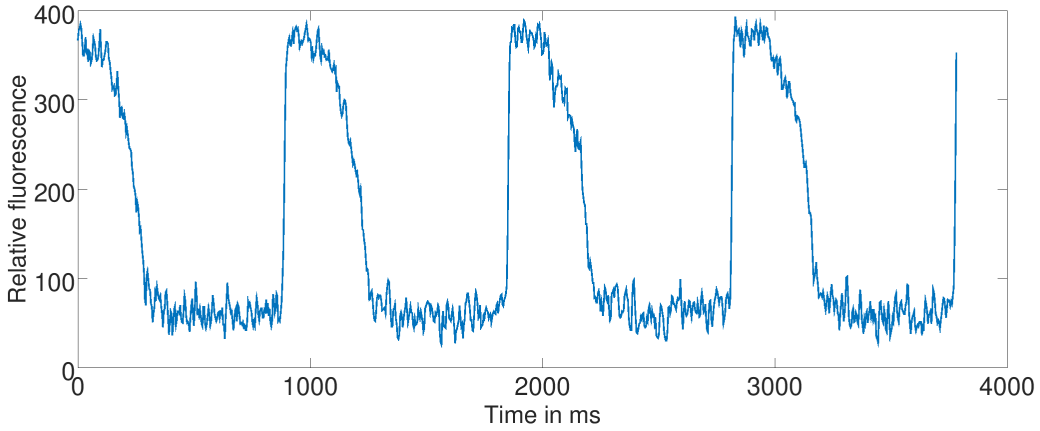



Figure 5.2: Inverted raw data

by taking the average of the values during a single recovery phase. Since the data is very noisy, automatically determining which values are to be considered to be in the recovery phase presents some challenges, so this is done manually. A representative recovery phase is identified by looking at the graphed, raw data (figure 5.2) and noting the approximate times at which the recovery phase starts and ends. A similar logic is used when finding the average maximum $amax$. The maximum is taken manually to be approximately the value at which each action potential peaks. Once the average extrema are found, the data is normalized by doing

$$\hat{u}_i(t_l) \leftarrow \frac{\hat{u}_i(t_l) - amin}{amax - amin}.$$

Lastly, it is convenient for the data to start with an upstroke, so the first part of most of the data sets is ignored (data acquisition does not always start in sync with the stimulation, it can start mid AP as in figure 5.2). To do so, the time $t_i^*$ when the first complete AP of a dataset starts is identified manually and the data is shifted by doing

$$\tilde{u}_i(t_l) \leftarrow \hat{u}_i(t_l - t_i^*).$$

These three transformations (inversion, normalization and shifting) take the raw data $\hat{u}_i$ and convert it into data $\tilde{u}_i$ that can be matched with the model using problem (3.5). However, one last operation must be done in order to be able to adequately compare $\tilde{u}_i$ with $u_i$. The trans-membrane potential $u_i(t, \tau)$ obtained by solving the model peaks at values that are less than 1. As explained in section 2.2.1, these peak values depend

on the parameters $\tau$ as well as the stimulating frequency. Since the values of $\tau$ change throughout the optimization process, these peak values change as well. The scaling factor $s_i$ scales $\tilde{u}_i$ so the experimental trans-membrane potential peaks at values different than 1. Leaving $s_i$ as a control variable permits the optimization algorithms to adjust $s_i$ with $\tau$ to account for the changes in peak values.

There is also a transformation to be done on $u_i(t,\tau)$. As explained in section 2.2.1, when considering multiple stimulations, the first few AP differ from the following ones. There is a transitory period before the solution stabilizes into a stable periodic response. The data is not always recorded from the start of the stimulations, so it may happen that the first recorded AP did not start from rest. Hence, to better match the data $\tilde{u}_i$, the following is done to disregard the first five AP of $u_i(t,\tau)$:

$$u_i(t,\tau) \leftarrow u_i(t - 5 \cdot BCL_i, \tau),$$

where $BCL_i$ is the basic cycle length chosen to match the stimulation pattern of $\tilde{u}_i$. More than five APs could have been disregarded in order to leave more time for the solution to stabilize, but five is deemed a sufficient amount of transitory APs and some computation time is saved.

Once the data and solution of the model are reformatted, they can be used to evaluate $J(\tau, S)$, which involves terms of the form $\int_0^{T_i^*} \mid u_i(t,\tau) - s_i\tilde{u}_i(t) \mid^2 dt$ and $\int_0^{T_i^*} \mid s_i\tilde{u}_i(t) \mid^2 dt$. Note that the upper bound on time is now $T_i^* = T_i - t_i^*$ since the data is shortened via a shifting by $t_i^*$. Since both $\tilde{u}_i$ and $u_i$ are only defined for certain discrete times, which may not coincide, evaluating these integral terms exactly is not possible. To approximate these terms, it is first necessary to have times at which the values of both $\tilde{u}_i$ and $u_i$ are known. Denoting the times where the value of $\tilde{u}_i$ is known by $\tilde{t}_l$, $l = 1, 2, \ldots, k_i$, $\tilde{t}_1 = 0$, $\tilde{t}_{k_i} = T_i^*$, the values $u_i(\tilde{t}_l, \tau)$ are required. To obtain these, the cubic spline of $u_i(t,\tau)$ is found as explained in section 2.3.3, using the points $(t_j, u_i(t_j,\tau))$ from the numerical solution of the model as knots of the spline. The spline is then evaluated at the times $\tilde{t}_l$ to find $u_i(\tilde{t}_l,\tau)$. Finally, the following approximations of the integrals are done, using the fact that each $\tilde{t}_l$ is evenly spaced:

$$\int_0^{T_i^*} \mid u_i(t,\tau) - s_i\tilde{u}_i(t) \mid^2 dt \approx \frac{T_i^*}{k_i} \sum_{l=1}^{k_i} (u_i(\tilde{t}_l,\tau) - s_i\tilde{u}_i(\tilde{t}_l))^2,$$

$$\int_0^{T_i^*} \mid s_i\tilde{u}_i(t) \mid^2 dt \approx \frac{T_i^*}{k_i} \sum_{l=1}^{k_i} (s_i\tilde{u}_i(\tilde{t}_l))^2.$$

If the $\tilde{t}_l$ were not evenly spaced, there would be a term inside the sums to account for the length of each interval $[\tilde{t}_l, \tilde{t}_{l+1}]$.

## 5.2 Verification fittings

As in chapter 4, a few remarks are in order. Once again, the *ode45* parameters given in (4.2) are used for all test in this chapter. Since multiple beats are the target of the optimization problem, $I_{stim}$ stimulates the cell periodically, using a BCL that is dependent on the target data. The value $v^*$ of $v$ at the time of stimulation is used when calculating $A$ in equation (2.9), as opposed to a fixed value of $v^*$ as in chapter 4. Also, $\beta = 0.25$ and $\Delta t = 2$ ms are used for $A$ and $I_{stim}$ in equations (2.9) and (2.10).

The Front-track Compass Search (FCS) method is used to solve the optimization problem (3.5) and is implemented the same way as in section 4.4 ($\delta = 0.02$, $c = 0.25$ and $w = 1.25$).

Some tests were conducted to verify if using FCS to solve problem (3.5)) is a process capable of identifying parameters that fit multiple APs. For these tests, three consecutive APs obtained by the MS model with given target parameters are used as the "experimental data". FCS is then fitting the model to itself, hence a perfect match is expected. At first, a single "dataset" is considered (i.e. $N = 1$ in (3.5)). The first tests results are shown in table 5.1. The target APs are obtained by solving the model with parameters $\tau^* = [0.3, 6, 130, 150]$, a first test case using $BCL = 600$ ms for $I_{stim}$ and a second test case using $BCL = 900$ ms. This target "dataset" is reformatted in a similar way as the $u_i(t, \tau)$ is; the first five APs are discarded, so the target APs are actually the 6th, 7th and 8th AP. Also, it follows that the optimal scaling factor is $S = 1$. The initial guess is $[\tau_0, S_0] = [0.4, 8, 100, 120, 1.1]$ for the 600 ms case and it is $[\tau_0, S_0] = [0.4, 8, 100, 120, 0.9]$ for the 900 ms case. Note that only the scaling factors differ in these initial guesses. The stopping criterion is first set to $fevals \geq 500$ and then it set to $fevals \geq 2000$.

Table 5.1: Verification of method: single BCL

| Target BCL (ms) | $fevals$ | $\tau_{final}$ | $S_{final}$ | $J(\tau_{final}, S_{final})$ |
|---|---|---|---|---|
| 600 | 502 | $[0.346, 6.34, 91.30, 152.10]$ | 0.993 | $1.868e-4$ |
| 900 | 509 | $[0.358, 6.56, 80.84, 155.91]$ | 0.990 | $3.931e-4$ |
| 600 | 2002 | $[0.327, 5.95, 86.20, 154.34]$ | 0.996 | $1.010e-6$ |
| 900 | 2008 | $[0.329, 5.95, 76.83, 159.11]$ | 0.991 | $4.99e-6$ |

The small value of the cost function and the quasi-superposed graphs of the target and the matched trans-membrane potentials (not shown) reveal that the fitting of the potential was successful in both cases. After 500 function evaluations, there is only a very small discrepancy seen between the target and the model potentials and after 2000 function evaluations, there is virtually no difference. However, it is worth noting that after 500 evaluations, while the fit is good in both cases, the two $\tau_{final}$ differ a bit from each other and differ a great amount from $\tau^*$, especially in $\tau_{open}$. As mentioned in chapter 2, this parameter controls the evolution of the recovery variable $v$ so it also affects the restitution properties during the recovery phase. Letting FCS do 2000 function evaluations instead of 500 did not remedy the problem. Keeping in mind that the goal of the parameter identification is to find a single set of parameters that fits multiple datasets, it is desirable for the two $\tau_{final}$ to be as close to $\tau^*$ as possible. This warrants the inclusion of more than one "dataset" in the cost function (i.e. $N \geq 2$), so that FCS finds $\tau_{final}$ that fit both (or more) datasets.

Table 5.2 shows the result of tests where both sets of APs, obtained from the solution of the model with $\tau^* = [0.3, 6, 130, 150]$ using $BCL = 600$ ms and $BCL = 900$ ms, are included in the cost function ($N = 2$). The initial guess is $[\tau_0, S_0] = [0.4, 8, 100, 120, 1.1, 0.9]$. The stopping criterion is $fevals \geq 500$ in the first case and $fevals \geq 2000$ in the second.

Table 5.2: Verification of method: multiple BCL

| Target BCL (ms) | $fevals$ | $\tau_{final}$ | $S_{final}$ | $J(\tau_{final}, S_{final})$ |
|---|---|---|---|---|
| $600 - 900$ | 507 | $[0.327, 6.54, 118.57, 147.43]$ | $[0.998, 0.990]$ | $6.751e-4$ |
| $600 - 900$ | 2010 | $[0.301, 6.01, 129.74, 150.07]$ | $[1.000, 1.000]$ | $1.097e-7$ |

Comparing the results in table 5.1 and table 5.2 shows that the multiple BCL fitting gives $\tau_{final}$ closer to $\tau^*$ after 500 function evaluations and $\tau_{final}$ almost exact after 2000 function evaluations (a maximal relative error of 0.34%). The $S_{final}$ are also correct.

To compare the value of the cost function from single BCL fittings to the value of the cost function for multiple BCL fittings, one can simply divide $J(\tau_{final}, S_{final})$ by $N$. It is thus seen that the fits are of similar quality after 500 evaluations when using a single BCL ($J = 3.931e-4$ for 900 ms case) and multiple BCL ($\frac{J}{2} = 3.376e-4$ for 600 ms$-$900 ms case). However, the fit is of significantly better quality after 2000 evaluations when using multiple BCL ($\frac{J}{2} = 5.485e-8$ for 600 ms$-$900 ms case) rather than a single BCL ($J = 1.01e-6$ for 600 ms case).

The verification fittings give satisfying results in the sense that the potential is very well fitted. However, the results suggest the need for multiple datasets to be included in the cost function if one hopes to find the right and unique $\tau_{final}$ matching data at different frequencies. Including multiple frequencies gives more information on the target restitution properties and thus better dictates the target value for $\tau_{open}$, which is the main parameter controlling recovery.

## 5.3 Single data fittings

For this thesis, six different potential recordings, acquired as explained in section 5.1, are considered. The datasets, labeled with numbers 1 through 6, are recordings from the same explanted swine heart stimulated at various frequencies. Table 5.3 shows the properties of each dataset.

Table 5.3: Properties of the six datasets

| Dataset | Heart rate (bpm/Hz) | BCL (ms) | Length of dataset $T_i^*$ (ms) |
|---|---|---|---|
| 2 | 59.17/0.99 | 1014 | 2534.5 |
| 1 | 62.37/1.04 | 962 | 2911.9 |
| 3 | 72.29/1.20 | 830 | 3215.3 |
| 4 | 74.81/1.25 | 802 | 3108 |
| 5 | 104.35/1.74 | 575 | 3082.1 |
| 6 | 145.63/2.43, 157.48/2.62 | 412, 381 | 3263.4 |

Dataset 6 has two different BCLs because of its observed stimulation pattern: The BCLs alternate between longer (412 ms) and shorter (381 ms) ones. When fitting the model to this dataset, $I_{stim}$ is adjusted to take this alternation into account. The reason for this alternation is unknown, but it is suspected that the increased stimulation frequency causes delays in the electric wave propagation to the left ventricle as described in chapter 1. The response of each individual cell is delayed due to the fact that it has not had time to recover entirely from the previous stimulation, so a sort of chain reaction from the pacing site to the actual recording site causes an alternation (large 31 ms gap) in the activation of the observed cell. This phenomenon is akin to the 2-2 behaviour that the MS model exhibits at high stimulating frequencies, but on the organ level instead of the cellular level.

Similarly to the verification process in section 5.2, the datasets are first fitted individually. In each case, one dataset is chosen to be the target data for the cost function (3.5) with $N = 1$. Table 5.4 shows the results of these fittings. The initial guess varies from dataset to dataset, a result of each case having its own tailored initial guess. The initial guesses are chosen according to results of preliminary tests, which are not shown. The stopping criterion varies from $fevals \geq 500$ to $fevals \geq 1000$.

Table 5.4: Single dataset fittings

| Dataset | $fevals$ | $[\tau_0, S_0]$ | $\tau_{final}$ | $S_{final}$ | $J(\tau_{final}, S_{final})$ |
|---|---|---|---|---|---|
| 2 | 1001 | $[0.3, 6, 130, 150, 0.95]$ | $[0.358, 9.24, 153.98, 177.95]$ | 0.96779 | $8.62e-3$ |
| 1 | 509 | $[0.5, 9, 200, 200, 0.90]$ | $[0.595, 10.83, 211.39, 198.44]$ | 0.91763 | $8.06e-3$ |
| 3 | 1002 | $[0.7, 8, 300, 300, 0.91]$ | $[0.632, 6.51, 287.48, 325.83]$ | 0.85545 | $1.20e-2$ |
| 4 | 1004 | $[0.55, 7, 215, 190, 0.87]$ | $[0.448, 5.31, 207.84, 202.22]$ | 0.87435 | $2.05e-2$ |
| 5 | 804 | $[0.6, 9, 80, 180, 0.89]$ | $[0.584, 8.86, 81.41, 168.39]$ | 0.91417 | $6.52e-3$ |
| 6 | 1010 | $[0.7, 4, 120, 300, 0.85]$ | $[0.700, 7.61, 48.16, 207.68]$ | 0.86944 | $8.65e-3$ |

It is expected that the value of $J$ be higher for the fitting of experimental data compared to the value of $J$ when fitting the model to itself for two reasons. The first reason is that the data is very noisy (see figure 5.2 for an example). The model cannot reproduce this noisy behaviour, so the measured potential and the potential given by the model will always somewhat differ. The second reason to expect a larger $J$ value is that there is a possibility that the model is simply unable to match a particular dataset. There might be an issue where the behaviour of the dataset is fundamentally different from the behaviour of the potential given by the model. The fitting process is thus adjusting $\tau$ and $S$ so that the discrepancy between the data and the trans-membrane potential is minimized, but it is impossible for $J$ to eventually reach 0.

The results in table 5.4 show that the fitting of dataset 5 is the best, followed closely by those of datasets 1 and 2, while the fits of datasets 3 and 4 are less successful. To illustrate the quality of fit for a smaller $J$ value compared to a larger one, figure 5.3 shows the results of fitting dataset 5 and dataset 4, respectively.

Even in the worst case (dataset 4), the fit is still good. A few areas are less well fitted for this case, such as the plateau phases of the first and third APs as well as the last two recovery phases. For dataset 5, there is no specific area that is a source of discrepancy.

In cases where the initial guess is "bad", such as for the fittings of datasets 2 and 6, FCS reduces the value of $J$ a great amount. For dataset 2, $J(\tau_0, S_0) = 0.177$ and

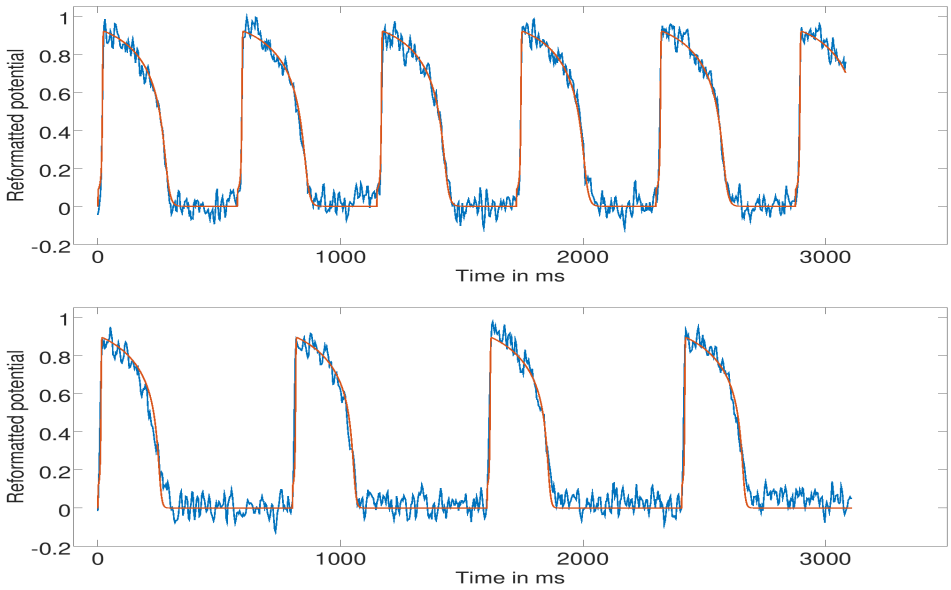



Figure 5.3: Results of fitting datasets individually. Top: reformatted dataset 5 (blue) and solution of the MS model with $\tau$ found by FCS stimulated at $BCL = 575$ ms (orange). Bottom: reformatted dataset 4 (blue) and solution of the MS model with $\tau$ found by FCS stimulated at $BCL = 802$ ms (orange)

$J(\tau_{final}, S_{final}) = 0.00862$. FCS thus reduces $J$ by a factor of roughly 20. For dataset 6, $J(\tau_0, S_0) = 0.404$ and $J(\tau_{final}, S_{final}) = 0.00865$. FCS thus reduces $J$ by a factor of roughly 47. Hence, the optimization method seems to perform well even with relatively bad initial guesses.

On the other hand, in cases where the initial guess is better, such as for the fitting of datasets 1 and 5, FCS still manages to reduce $J$ enough to get a great fit. For dataset 1, $J(\tau_0, S_0) = 0.0141$ and $J(\tau_{final}, S_{final}) = 0.00806$. FCS thus reduces $J$ by a factor of roughly 1.75. For dataset 5, $J(\tau_0, S_0) = 0.0145$ and $J(\tau_{final}, S_{final}) = 0.00652$. FCS thus reduces $J$ by a factor of roughly 2.2. These factors seem small compared to the those for datasets 2 and 6, but the initial guesses were already very good, so there was less progress to be made. Still, FCS manages to find parameters that fit the data as well or in a better way than for the cases where the reduction factor is large.

One last remark is that the $\tau_{final}$ vary immensely from dataset to dataset. As mentioned before, it is desirable to find a single $\tau$ that fits all datasets, which is not the case for these $\tau_{final}$. Fitting a single dataset is not enough to guarantee that the model will fit other datasets with different frequencies. To illustrate this fact, figure 5.4 compares

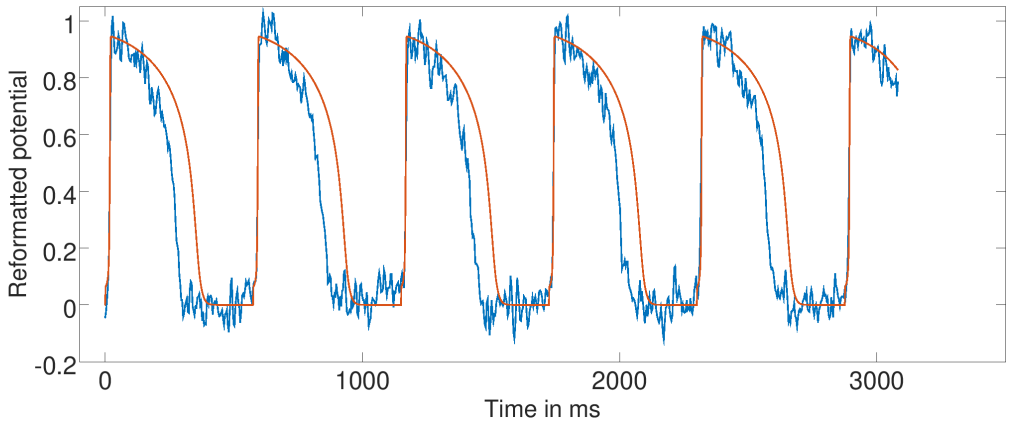



Figure 5.4: Reformatted dataset 5 (blue) and solution of MS model with $\tau_{final}$ fitted with dataset 2 (orange).

dataset 5 to the solution of the model stimulated using the $BCL = 575$, but using $\tau_{final}$ that match dataset 2. $S = 0.945$ is manually chosen to better match the peaks of the data and the solution of the model, since $S_{final} = 0.968$ is the scaling factor for dataset 2, not 5. Clearly, this $\tau_{final}$ does not yield a potential that fits dataset 5. The next step, as in the verification process, is then to fit multiple datasets at once.

## 5.4 Multiple data fittings

For the next fittings, the cost function (3.5) includes $N = 3$ chosen datasets ordered from largest to smallest BCL. Table 5.5 shows results for five such fittings. Again, the initial guess varies from case to case as it is chosen based on the $\tau_{final}$ in table 5.4. For example, for the fitting of datasets 2-1-3, $\tau_{in,0} = 0.5$ is chosen since $\tau_{in,final}$ varied between 0.358 and 0.632 for the fittings of 2, 1 and 3 individually. The stopping criterion is $fevals \geq 1000$.

The values of $J(\tau_{final}, S_{final})$ show that the fitting of datasets 2-1-3 is the best, being slightly better than the 4-5-6, 2-1-6 and 1-3-6 cases, and the fitting of 2-4-5 is the worst, having a $J$ value more than twice as large as any other case.

As before, to compare multiple dataset fitting with single dataset ones, $J(\tau_{final}, S_{final})$ can be divided by $N = 3$. For the 2-1-3 case, $\frac{J}{3} = 0.0174$, which is a similar value as for the worst single dataset fitting. Another comparison is to add up the three $J(\tau_{final}, S_{final})$

Table 5.5: Triple dataset fittings

| Datasets | $[\tau_0, S_0]$ | $\tau_{final}$ | $S_{final}$ | $J(\tau_{final}, S_{final})$ |
|---|---|---|---|---|
| $2-1-3$ | $[0.5, 9, 200, 200, 0.95, 0.90, 0.85]$ | $[0.500, 9.36, 538.54, 244.59]$ | $[0.890, 0.913, 0.923]$ | $5.23e-2$ |
| $4-5-6$ | $[0.6, 5, 90, 250, 0.86, 0.84, 0.83]$ | $[0.651, 6.32, 53.66, 245.33]$ | $[0.877, 0.857, 0.872]$ | $6.09e-2$ |
| $2-4-5$ | $[0.4, 7, 120, 150, 0.92, 0.89, 0.86]$ | $[0.405, 8.04, 612.26, 239.70]$ | $[0.889, 1.068, 0.834]$ | $1.68e-1$ |
| $2-1-6$ | $[0.5, 8, 150, 200, 0.92, 0.89, 0.86]$ | $[0.535, 9.06, 251.15, 224.97]$ | $[0.906, 0.936, 0.822]$ | $7.02e-2$ |
| $1-3-6$ | $[0.65, 8, 170, 200, 0.89, 0.88, 0.86]$ | $[0.621, 9.04, 183.05, 219.20]$ | $[0.884, 0.927, 0.827]$ | $7.72e-2$ |

for the fittings of 2, 1 and 3 individually, which gives 0.0287 (roughly half of $J = 0.0523$ for case 2-1-3). For the 2-4-5 case, $\frac{J}{3} = 0.056$, which is much worse than any of the single dataset fittings. Adding up the three $J$ values for the individual fittings of 2, 4 and 5 gives 0.0356 (roughly one sixth of $J = 0.168$ for case 2-4-5).

As expected, fitting multiple datasets at once is more challenging, with fittings being roughly 2 to 6 times worse statistically. Figures 5.5 and 5.6 illustrate the quality of the fits graphically for the best (2-1-3) and average (2-1-6) multiple data fittings. These two cases share two datasets (2 and 1). Comparing the top and middle graphs of figures 5.5 and 5.6 shows that datasets 2 and 1 are as well fitted in case 2-1-3 as in case 2-1-6. The difference in the values of $J$ for these two cases seems to come from the difference in fit of the third dataset (dataset 3 in one case and dataset 6 in the other). Graphically, it is observable that dataset 3 is better fitted than dataset 6, especially since the fit of dataset 6 deteriorates after the first few APs. This highlights a trend for the five fittings considered in table 5.5: cases where the range of the BCLs is smaller (2-1-3 and 4-5-6) lead to better fits than cases where the BCLs vary more (2-4-5, 2-1-6 and 1-3-6).

The next tests attempt to fit all six datasets simultaneously ($N = 6$ in (3.5)). Table 5.6 shows the results for 3 such tests. The initial guess varies from test to test in an attempt to obtain better results ($[\tau_0, S_0] = [0.5, 8, 200, 225, 0.89, 0.91, 0.92, 0.90, 0.85, 0.84]$ for the first and third test, and $[\tau_0, S_0] = [0.3, 6, 130, 150, 0.93, 0.92, 0.91, 0.89, 0.86, 0.84]$ for the second). The first two tests are done with FCS while the third is done with HCS (using search parameters $\delta = 0.2$ and $c = 0.85$ as in section 4.4). The stopping criterion is $fevals \geq 2000$.

Once again, the results in table 5.6 are compared to previous tests. For the first fit with FCS, $J(\tau_{final}, S_{final}) = 0.273$, so $\frac{J}{6} = 0.0455$ which is a value similar to the value $\frac{J}{3}$ for the worst of the triple data fittings. Simply put, the fit is not very good. With similar $J$ values, the two other test did not fare much better, i.e. changing the initial

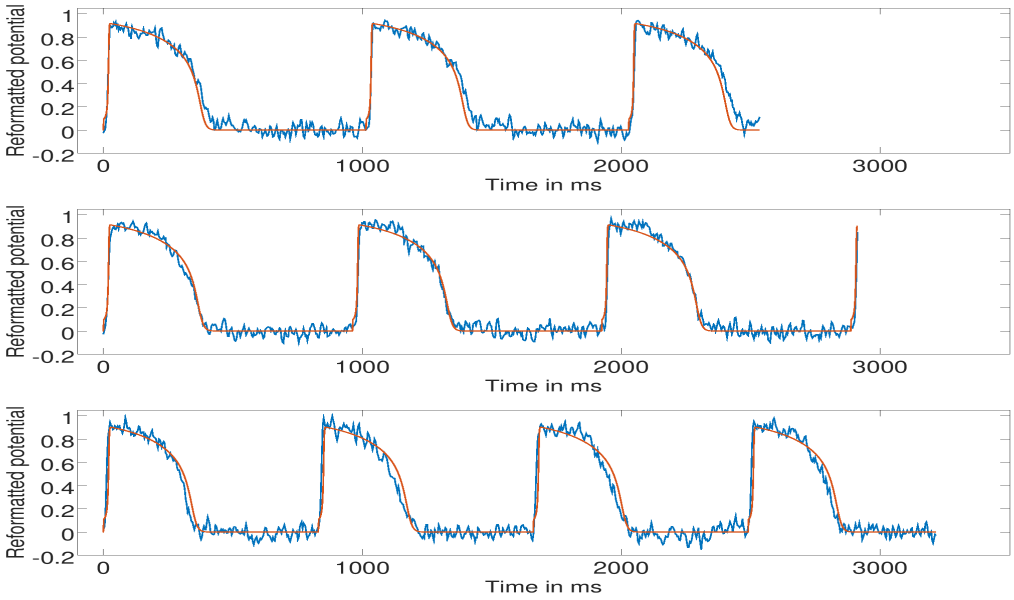



Figure 5.5: Results of fitting datasets 2-1-3. Top: reformatted dataset 2 (blue). Middle: reformatted dataset 1 (blue). Bottom: reformatted dataset 3 (blue). All: solution of the MS model with $\tau$ found by FCS (orange)

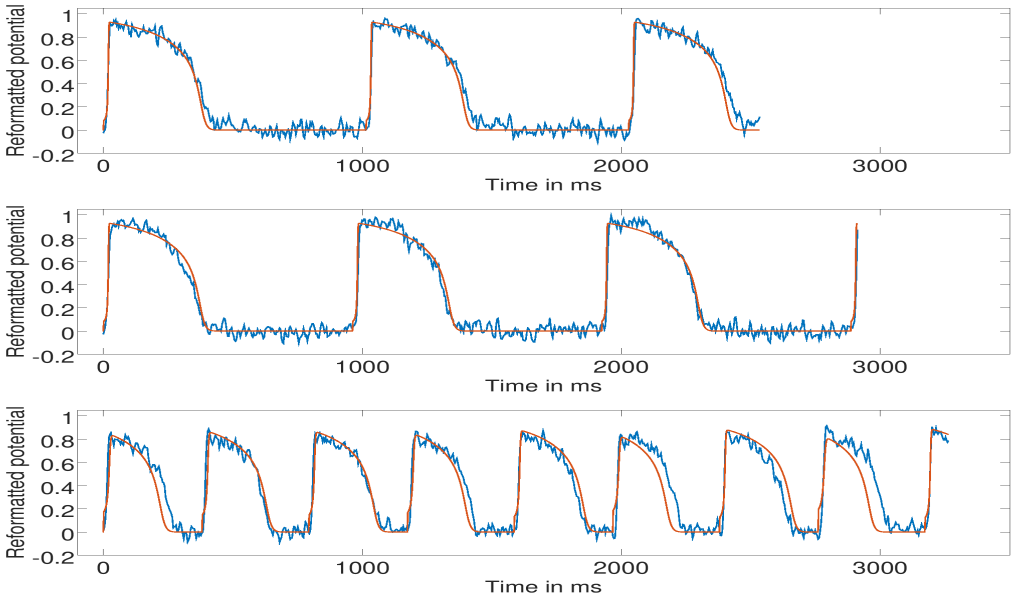



Figure 5.6: Results of fitting datasets 2-1-6. Top: reformatted dataset 2 (blue). Middle: reformatted dataset 1 (blue). Bottom: reformatted dataset 6 (blue). All: solution of the MS model with $\tau$ found by FCS, stimulated at corresponding BCL (orange)

Table 5.6: Fittings of all six datasets

| Method | $\tau_{final}$ | $S_{final}$ | $J(\tau_{final}, S_{final})$ |
|---|---|---|---|
| FCS | $[0.500, 8.45, 298.79, 215.05]$ | $[0.884, 0.905, 0.932, 1.101, 0.893, 0.796]$ | 0.273 |
| FCS | $[0.239, 7.54, 392.88, 156.96]$ | $[0.935, 0.957, 0.980, 1.156, 0.963, 0.872]$ | 0.289 |
| HCS | $[0.528, 8.385, 288.02, 223.54]$ | $[0.879, 0.899, 0.926, 1.093, 0.886, 0.793]$ | 0.273 |

guess or changing the optimization method does not help much.

Figure 5.7 show the fits of all six datasets for the third test of table 5.6. The fit looks good for datasets 1, 3 and 5, with only small discrepancies being openly visible. The fit is average for dataset 2 (the repolarization phases are shorter for the model than for the data). The fit for dataset 6 is a bit worse, with several mismatches between the plateau and repolarization phases being visible. Finally, the fit of dataset 4 is very bad. A closer investigation of each of the $J_i(\tau_{final}, s_{i,final})$ for each of the datasets reveals that $J_1 = 0.0381$, $J_2 = 0.0195$, $J_3 = 0.0191$, $J_4 = 0.141$, $J_5 = 0.0076$ and $J_6 = 0.0483$. Comparing these $J_i$ to the results in table 5.4 confirms the above assessment of the quality of the fits. It is also noticeable that $J_4$ is contributing more than half of the total $J$, so the poor performance of the algorithm can be mostly explained by the bad fit of dataset 4.

One might suggest decreasing the scaling factor $s_4$ as a way of improving the fit, since figure 5.7 clearly shows that the data peaks much higher than the trans-membrane potential. While this modification would yield a better match in terms of peak potential, the discrepancy between the data and the model would increase for the plateau and repolarization phases because the potential given by the model would start decreasing far later than the data. It must be trusted that the $s_4$ found by HCS is the best choice of scaling factor for the $\tau_{final}$.

As with the single data fittings, it is investigated whether using $\tau_{final}$ obtained for some datasets yields a good fit for different datasets. Table 5.7 shows the values of the cost function for all six datasets when using $(\tau_{final}, S_{final})$ obtained for triple dataset fittings. For example, the first row shows the value of $J$ for all six datasets using the $\tau_{final}$ from the source case, fitting 2-1-3 (the missing three scaling factors $s_4$, $s_5$, and $s_6$ are chosen manually to best match the peaks of the APs). The manually chosen scaling factors are in bold.

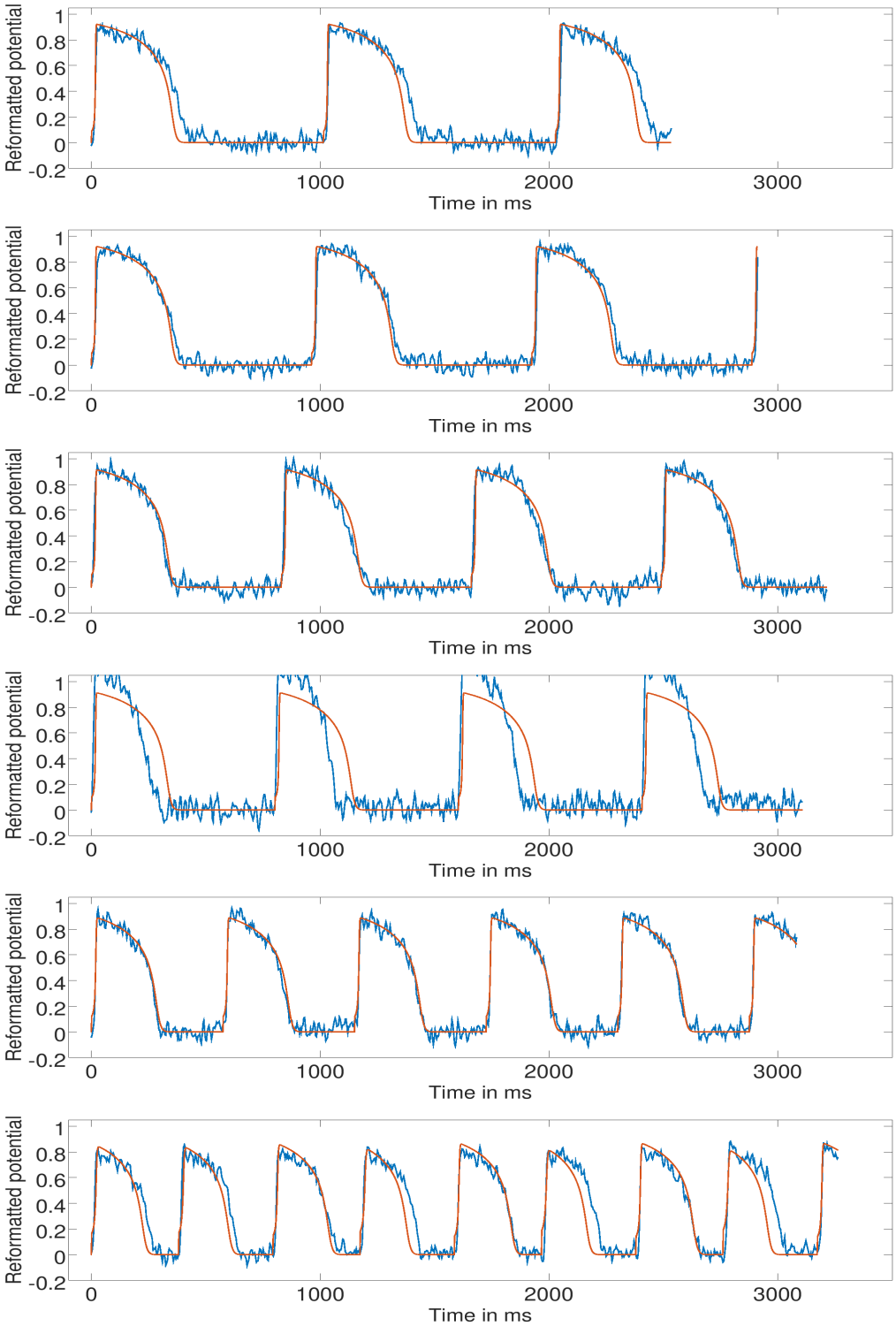



Figure 5.7: Results of fitting datasets 2-1-3-4-5-6. Top to bottom: reformatted datasets 2, 1, 3, 4, 5 and 6, respectively (blue). All: solution of the MS model with $\tau$ found by HCS, stimulated at corresponding BCL (orange)

Table 5.7: Fit values of all six datasets using parameters obtained for three datasets

| Source case | $\tau$ | $S$ | $J(\tau, S)$ |
|---|---|---|---|
| 2-1-3 | $[0.500, 9.36, 538.54, 244.59]$ | $[0.890, 0.913, 0.923, \mathbf{0.890, 0.857, 0.800}]$ | 0.344 |
| 4-5-6 | $[0.651, 6.32, 53.66, 245.33]$ | $[\mathbf{0.870, 0.870, 0.870}, 0.877, 0.857, 0.872]$ | 0.541 |
| 2-4-5 | $[0.405, 8.04, 612.26, 239.70]$ | $[0.889, \mathbf{0.910, 0.910}, 1.068, 0.834, \mathbf{0.800}]$ | 0.301 |
| 2-1-6 | $[0.535, 9.06, 251.15, 224.97]$ | $[0.906, 0.936, \mathbf{0.921, 0.919, 0.890}, 0.822]$ | 0.398 |
| 1-3-6 | $[0.621, 9.04, 183.05, 219.20]$ | $[\mathbf{0.918}, 0.884, 0.927, \mathbf{0.914, 0.890}, 0.827]$ | 0.330 |

From the results in table 5.7, it is seen that in some cases, the value of the fit on all six datasets is similar whether all six datasets are included in the cost function ($J = 0.273$) or only three are ($J = 0.301$ for 2-4-5 case).

Because the scaling factors in bold are chosen manually so that the peaks of the potentials match the data, it is expected that there exists a better choice of scaling factors for the $\tau_{final}$ that could reduce $J$. This is a phenomenon similar to the one highlighted previously for the scaling factor $s_4$ of the six dataset fitting. The point is that the $J$ values in table 5.7 could be a bit lower if the optimal scaling factors were chosen.

The previous remark suggests that the fit for all six datasets is indeed essentially of the same quality with either $N = 3$ or $N = 6$ datasets included in the cost function. If that is the case, including many datasets rather than a few would not be worth the increased computational time due to the larger number of solutions computed for each cost function evaluation and the larger number of directions to check at each iteration of FCS, since more scaling factors are needed.

However, it is worth considering that perhaps the value of the cost function might not be the only indicator of a good fit. As is seen with many of the previous tests, the values for $\tau$ in table 5.7 vary substantially, especially $\tau_{open}$ which varies from 53.66 to 612.26. Different $\tau$ values can thus give similar cost function values, so identifying the correct $\tau$ is difficult. Strategically choosing the pacing frequencies for the target data is something that could help, but further investigation is required on the optimal choice of representative frequencies as it is difficult to extract definitive conclusions from the small sample size of tests presented here.

# Conclusion

In this thesis, four different non-differentiable optimization methods are presented, of which three are original variants of the standard Compass Search (CS) method. The methods are tested and compared with each other by applying them to two optimization problems with applications in cardiac electrophysiology. The results of these tests led to the following conclusions about the four algorithms. The Front-track Compass Search and Hybrid Compass Search methods represented a significant improvement in performance when compared to the standard Compass Search method in this context. The Golden Compass Search method, while not vastly better than CS, still performed adequately, thus providing a total of three new non-differentiable optimization methods. The non-standard implementation of the CS methods (compared to the article from which CS is taken [13]) also proved useful in the electrophysiological context. A comparison of the performance of the CS methods with the Nelder-Mead method as used in [24] shows that the CS methods are capable of finding minimizers of similar or far superior quality in a comparable amount of time/function evaluations. If anything, the CS methods performed more reliably, obtaining very good minimizers in every test case.

An interesting avenue for future research is to study the CS methods, particularly the three original variations, in a more general way. Applying them to a broader set of optimization problems and going more in depth in the study of various effects like varying the search parameters, changing the implementation, such as changing the generating set $D$ as is proposed in this thesis, or the sensibility to the initial guess could help further determine the viability of the CS methods.

When used in the electrophysiological context, the CS methods paired with the choice of cost functions gave good results. For the first problem consisting in matching phase durations, the optimization methods worked very well and consistently found parameters for which the MS model matches phase durations to within 1% or less in most cases.

When considering the second problem (matching the potential directly), it is seen that the primary optimization method used, FCS, is capable of identifying parameters

and scaling factors that make the MS model fit optical imaging data reasonably well. For single stimulation frequencies, the fit of the trans-membrane potential to the noisy data is very good, which is corroborated by the small value of the cost function and the small visual discrepancy observed when the potential and the data are graphed together. When trying to fit data from multiple stimulation frequencies, the quality of the fit deteriorates as the complexity of the problem is increased, but the resulting fits are still reasonable, with most datasets being well fitted while only a few datasets suffer from an sub-optimal fit. The fit of the datasets could perhaps be improved, but before suggesting that it is the methods that are ill-equipped to perfect the fit, a word must be said on the data that is being fitted. The optical imaging used for data acquisition does present some drawbacks when used to fit single-cell models like the MS model. The resolution of the camera does not permit the acquisition of single-cell data as it captures the electrical activity of a small area of tissue that contains a multitude of cells. The behaviour of tissue differs from the behaviour of single cells, so it is perhaps impossible for a single cell model to replicate the tissue data. Also, the lack of appropriate scaling for the data forces extra pre-processing of the data before matching models. Moreover, there is an increase in complexity, due to the inclusion of scaling factors, that can hinder the quality of the fit. Nonetheless, the inclusion of the scaling factors as control variables seemed to be a good way of fitting the MS model to the optical data.

One last remark on the performance of FCS for the fitting of the trans-membrane potential is that FCS did a reasonably good job while only requiring roughly 1000 function evaluations. Also worth noting is that, in some cases, most of the progress was observed in the first 500 evaluations. Compared to genetic algorithms, which can require thousands and thousands of evaluations (10000+ evaluations over 100 generations in [32]), FCS offers a less computationally costly alternative for problems in which few parameters must be identified, given that a suitable initial guess can be found.

Future work on recorded potential fitting could include using direct, single-cell potential recordings with the FCS method to eliminate the need for scaling factors, trying to match more sophisticated models to the optical data or investigating further into the development and applications of better optimization algorithms, eventually leading to global minimizers.

# Bibliography


[1] U.M. Ascher and C. Greif. *A First Course in Numerical Methods*, volume 1. SIAM, 2011.

[2] G. W. Beeler and H. Reuter. Reconstruction of the action potential of ventricular myocardial fibres. *The Journal of Physiology*, 268(1):177–210, 1977.

[3] E.K.P Chong and S.H. Żak. *An Introduction to Optimization*, volume 1. Wiley-Interscience, second edition, 2001.

[4] C. Davis. Theory of positive linear dependance. *American Journal of Mathematics*, 76(4):733–746, 1954.

[5] J.R. Dormand and P.J. Prince. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.

[6] R. FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, 1:445–466, 1961.

[7] Source Forge. Octave forge: integrate adaptive.m. https://sourceforge.net/p/octave/odepkg/ci/5e10fde2a5e64d76997f579466007cfc0f0091b0/tree/inst/integrate_functions/integrate_adaptive.m. Accessed: 2019-05-09.

[8] P.C. Franzone, L.F. Pavarino, and S. Scacchi. *Mathematical Cardiac Electrophysiology*, volume 13. Springer International Publishing, first edition, 2014.

[9] W. Groenendaal, F.A. Ortega, A.R. Kherloplan, A.C. Zygmunt, T. Krogh-Madsen, and D.J. Christini. Cell-specific cardiac electrophysiology models. PLOS Computational Biology https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004242. 22 pages (2015) Accessed: 2019-07-04.

[10] A.L. Hodgkin and A.F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544, 1952.

[11] J. Kaur, A. Nygren, and E.J. Vigmond. Fitting membrane resistance along with action potential shape in cardiac myocytes improves convergence: application of a multi-objective parallel genetic algorithm. PLOS ONE 9(9) https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0107984. 10 pages (2014) Accessed: 2019-07-04.

[12] J. Keener and J. Sneyd. *Mathematical Physiology: Systems Physiology*, volume 8. Springer, second edition, 2009.

[13] T.G. Kolda, R.M Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482, 2003.

[14] D.M. Lombardo, F.H. Fenton, S.M. Narayan, and W.J. Rappel. Comparison of detailed and simplified models of human atrial myocytes to recapitulate patient specific properties. PLOS Computational biology https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005060. 15 pages (2016) Accessed: 2019-07-09.

[15] Mathworks. odeset. https://www.mathworks.com/help/matlab/ref/odeset.html#bu2m9z6-2. Accessed: 2019-05-09.

[16] C.C. Mitchell and D.G. Schaeffer. A two-current model for the dynamics of cardiac membrane. *Bulletin of Mathematical Biology*, 65(5):767–793, 2003.

[17] J. Nagumo, S. Arimoto, and S. Yoshizawa. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10):2061–2070, 1962.

[18] J.A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[19] J. Nocedal and S.J. Wright. *Numerical Optimization*, volume 1. Springer, second edition, 2006.

[20] Octave.org. https://www.gnu.org/software/octave/. Accessed: 2019-07-09.

[21] Octave.org. 24.1.1 matlab-compatible solvers. https://octave.org/doc/interpreter/Matlab_002dcompatible-solvers.html. Accessed: 2019-05-09.

[22] Octave.org. 29.1 one-dimensional interpolation. https://octave.org/doc/v4.0.1/One_002ddimensional-Interpolation.html. Accessed: 2019-05-09.

[23] Government of Canada. Heart disease in canada. https://www.canada.ca/en/public-health/services/publications/diseases-conditions/heart-disease-canada.html. Accessed: 2019-07-02.

[24] D.V. Pongui Ngoma. *Identification des paramètres dans les modèles ioniques en électrophysiologie cardiaque [Parameter identification in ionic models for cardiac electrophysiology]*. PhD thesis, Université Marien Ngouabi, 2016.

[25] M. Pop, M. Sermesant, D. Lepiller, M.V. Truong, E.R. McVeigh, E. Crytal, A. Dick, H. Delingette, N. Ayache, and G.A. Wright. Fusion of optical imaging and MRI for the evaluation and adjustment of macroscopic models of cardiac electrophysiology: a feasibility study. *Med Image Anal*, 13(2):370–380, 2009.

[26] J. Relan, P. Chinchapatnam, M. Sermesant, K. Rhode, M. Ginks, H. Delingette, C.A. Rinaldi, R. Razavi, and N. Ayache. Coupled personalization of cardiac electrophysiology models for prediction of ischaemic ventricular tachycardia. *Interface Focus*, 1(3):396–407, 2011.

[27] J. Relan, M. Pop, H. Delingette, GA. Wright, N. Ayache, and M. Sermesant. Personalization of a cardiac electrophysiology model using optical mapping and mri for prediction of changes with pacing. *IEEE Trans. Biomed. Eng.*, 58(12):3339–3349, 2011.

[28] M. Rioux. *Numerical computations of action potentials for theheart-torso coupling problem*. PhD thesis, University of Ottawa, 2011.

[29] M. Rioux and Y. Bourgault. A predictive method allowing the use of a single ionic model in numerical cardiac electrophysiology. *ESAIM: Mathematical Modelling and Numerical Analysis*, 47(4):987–1016, 2013.

[30] L.F. Shampine and M.W. Reichlet. The Matlab ODE suite. *SIAM Journal on Scientific Computing*, 18(1):1–22, 1997.

[31] J. Sundnes, G.T. Lines, X. Cai, B.F. Nielsen, K.A. Mardal, and A. Tveito. *Computing the Electrical Activity in the Heart*, volume 1. Springer Berlin Heidelberg, first edition, 2006.

[32] Z. Syed, E. Vigmond, S. Nattel, and L.J. Leon. Atrial cell action potential parameter fitting using genetic algorithms. *Medical and Biological Engineering and Computing*, 43(5):561–571, 2005.

[33] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25, 1997.