Modelling and Simulating Retinal Dynamics and Physiology

by

Belal Abuelnasr

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Mathematics
University of Toronto

Modelling and Simulating Retinal Dynamics and Physiology

Belal Abuelnasr
Doctor of Philosophy

Graduate Department of Mathematics
University of Toronto
2024

Abstract

We present a detailed physiological model of the (human) retina that includes the biochemistry and electrophysiology of phototransduction, neuronal electrical coupling, and the spherical geometry of the eye. The model is a parabolic-elliptic system of partial differential equations based on the mathematical framework of the bi-domain equations, which we have generalized to account for multiple cell-types. We discretize in space with non-uniform finite differences and step through time with a custom adaptive time-stepper that employs a backward differentiation formula and an inexact Newton method. A refinement study confirms the accuracy and efficiency of our numerical method. We generalize our time-stepping scheme to higher order and derive estimates for the corresponding local truncation errors. Numerical simulations using the model compare favourably with experimental findings, such as desensitization to light stimuli and calcium buffering in photoreceptors. Other numerical simulations suggest an interplay between photoreceptor gap junctions and inner segment, but not outer segment, calcium concentration. Applications of this model and simulation include analysis of retinal calcium imaging experiments, the design of electroretinograms, the design of visual prosthetics, and studies of ephaptic coupling within the retina.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The retina is a unique part of the brain in that it is physically exposed to the outside world and has a relatively simple anatomical structure as compared to other parts of the brain [35]. This explains why it has been heavily studied for well over a century now [35] and, hence, much about retinal physiology and anatomical structure is well understood. For example, the anatomical structure and density of photoreceptors, which are the site of light detection and where signal transduction and processing begins [98], are well known in numerous species [127, 58, 45, 134]. Moreover, much has been discovered about the various processes and functions of the retina. For instance, phototransduction, which is the process in which light is converted into an electrical signal, is understood in great detail. There is a widely accepted description of phototransduction including the various molecules and proteins crucial to this process, such as opsins that isomerize after absorbing a photon and guanosine 3',5'-cyclic monophosphate (cGMP) that keeps some cGMP-gated cation channels open maintaining a dark current [44]. As a result of all this research, many biological models of the various retinal neurons have been developed [64, 124, 123, 7, 59].

Being a physically accessible part of the brain, there have been numerous experimental studies on the electrical activity in and around the retina as a result of light

stimuli. Visual prosthetics [41, 25, 126] and medical diagnosis [26, 35] are two prominent applications of such studies. Electroretinograms (ERGs) are a clinical diagnosis tool in which patients are shown various flashes of light stimuli and the resulting change in electrical potential on the eye surface is recorded. ERGs can be used to detect various diseases, such as retinitis pigmentosa and retinal vascular diseases [26]. A biologically-realistic, full-retina model would aid such studies and their applications.

Considering that the retina is composed of hundreds of millions of neurons, we will use a continuum model in which the electrical activity is described in spatial aggregate rather than in a cell-by-cell basis. The bi-domain model, first introduced by Tung, is such a continuum model, which uses homogenization to account for the multiple scales present in similar tissue-level models [120]. This model includes two separate domains (extracellular and intracellular spaces) which occupy the same physical space. Current is allowed to flow between the two domains (see Section 2.2.1). The currents and the dynamic variables in the bi-domain equations are spatial averages, over a large number of cells, of the corresponding currents and dynamic variables for individual cells, derived through homogenization [120, 55]. Homogenization is a mathematical technique which is used to obtain the macroscopic properties of a system from its microscopic ones [55, 23]. It can be used to obtain averaged equations from a system of partial differential equations whose (spatial) domain has a periodic microstructure [66]. See [66, 55] for a derivation of the bi-domain equations using homogenization.

The bi-domain equations have been applied extensively in modelling cardiac tissue. One such example is in predicting and suggesting mechanisms [109, 99, 102, 100, 101, 12] for cardiac strength-interval curves [65]. Results from studies using the bi-domain equations [101, 12] resembled those obtained from experimental measurements [30, 84]. In some cases, studies using the bi-domain equations [109] predicted mechanisms that were only later confirmed by experimental studies [129, 69, 88, 121, 89, 128,

68]. We hope to attain similar results in modelling the retina using the bi-domain equations.

The bi-domain equations have also been used to study neural tissues [87, 106, 107, 83]. The most notable bi-domain model of the retina was proposed by Dokos et al. [34], which is concerned with epiretinal stimulation via stimulus electrodes. This model has since been further developed, in various directions, for various purposes, such as handling different electrode stimulation techniques, using finite element implementation, and incorporating more details of the retina [63, 131, 132, 62, 110, 1, 5]. Numerous studies have been conducted based on the various versions of this model and have largely been concerned with visual prosthetics and/or electrical stimulation of the retina [2, 111, 112, 4, 6]. Given the scope of their work, the model proposed by Dokos et al. does not accurately describe the entire geometry of the retina and makes no attempt to model the entirety of the vitreous chamber. Also, the stimuli to the retina in most versions of this model was provided solely by stimulus electrodes. Only one version of this model was developed with light stimuli, but without detailed biological descriptions of some of the neurons, including photoreceptors, nor did they account for the geometry of the eye [132]. In particular, they studied the retinal response to small and large light-spot stimuli. Their findings were consistent with experimental studies, especially as it pertains to the relation between the size of the stimuli and surround antagonism [132].

An important reason for using the bi-domain equations in modelling the retina (and cardiac tissue) is that it provides an accurate description of the micro-scale structures in a macro-scale model. To illustrate, tissues are made out of cells on the micro-scale, and the intracellular and extracellular spaces are physically separated by cell membranes. Using the bi-domain equations, we retain this micro-scale description in the macro-scale model. However, when modelling tissues with multiple, densely packed cell-types, such as the four known photoreceptors of the retina [35, 98, 72], the

presence of the different intracellular spaces is not addressed in the macro-scale model. For this reason, we generalized our model to handle this multi-domain scenario (see Section 2.2.3 for the derivation of the multi-domain equations). This generalization also allows us to apply different light stimuli for the different photoreceptors. Thus, we are able to account for the differences in the sensitivities of the various photoreceptors to light of certain wavelengths [35, 98, 72]. A dissimilar multi-domain model of the retina, based on Dokos et al., has been proposed [5]. The multiple domains in that model represent the different compartments of the retinal ganglion cells, such as the dendrites, soma and axon, rather than different cell-types [5, 4, 6].

We present a detailed model of the human retina, which takes into account retinal physiology and the spherical geometry of the eye. While we present our model in the human species, the model and its framework is readily generalizable to other species (see Chapter 2). Light incident on the retina provides the stimuli, through a model of the phototransduction pathway [64]. The retina model is a system of PDEs which we solve using a finite difference scheme (see Sections 2.2.1 and 4.1). We overcome many challenges to successfully model the retina in this way including the 3D nature of this problem, the spherical geometry of the eye, numerical stiffness of the retinal dynamics, and the multiple space and time scales involved in this model. We implicitly step through time using a backward differentiation formula and Newton's method (see Section 4.2). We present an adaptive time-stepper (Section 4.2.1) and two inexact Newton methods (Section 4.3) that we were able to use, separately, to mitigate the computational cost and time arising from such complications. We generalize our adaptive time-stepping scheme to apply to higher order backward differentiation formulas (Chapter 5).

We discuss the details of our model in Chapter 2. We begin by describing the spatial aspect of the model (Section 2.1). We then discuss the mathematical basis of the model, including a derivation of the bi-domain and multi-domain equations and

the additional assumptions we make (Section 2.2). In an effort to be self-contained, we include a comprehensive overview of iterative methods (Chapter 3) including some of their preconditioning techniques (Section 3.3). While most of the contents of that chapter are not new, the presentation and some of the presented proofs are novel. This adequately prepares us for the detailed discussion of the numerical methods we used to solve our system of PDEs (Chapter 4). Details from the spatial discretization (Section 4.1) and the implicit time-stepping scheme used (Section 4.2), to the adaptive time-stepper (Section 4.2.1) and the two inexact Newton methods used (Section 4.3) are included in that discussion. We then generalize our adaptive time-stepper to be of higher order (Chapter 5). This generalization includes a novel derivation of the backward differentiation formulas coefficients (Section 5.1). Subsequently, we present and discuss findings (Chapter 6) obtained using a few numerical simulations of the model (Appendix A).

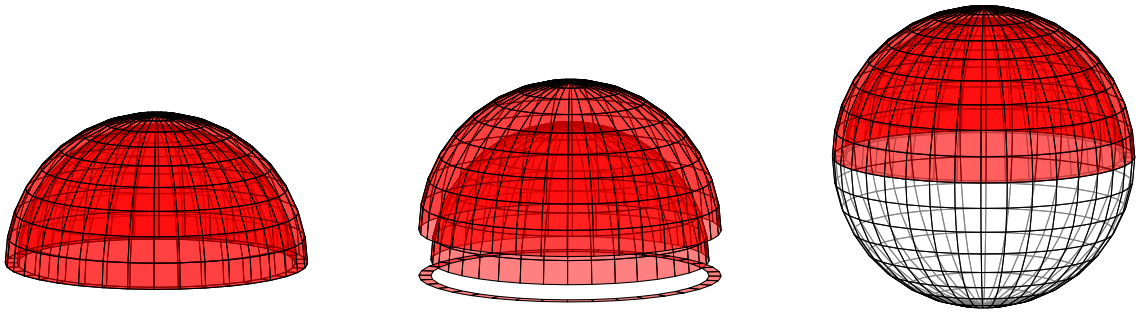
Chapter 2

Mathematical Model

In this chapter, we introduce the mathematical framework of our model, which has been published in Mathematical Biosciences [3]. This includes deriving the bi-domain equations and the resulting partial differential equation (PDE) system for the entire eye. We also generalize the bi-domain equations to the multi-domain equations to account for more cellular domains.

2.1 Geometrical Setup

We assume the eye $\mathcal{S} = \overline{B(0, r_{\text{eye}})} \subset \mathbb{R}^3$ to be a closed ball centered at the origin with radius $r_{\text{eye}} = 12.25$ mm [71]. We orient the eye so that the retina, $\mathcal{R} = \{(r, \theta, \varphi) \in \mathcal{S} : r_{\text{eye}} - r_{\text{retina}} \leq r \leq r_{\text{eye}},\ \varphi_{\text{retina}} \leq \varphi \leq \frac{\pi}{2}\}$, is situated on the north pole of $\mathcal{S}$. We choose $\varphi_{\text{retina}} = 0$ rad to obtain an experimentally accepted (outer) retinal radius value of 19.2 mm [73]. We segment the retinal boundary, $\partial\mathcal{R}$, into an outer boundary, $\partial\mathcal{R}_{\text{o}}$, lateral boundary, $\partial\mathcal{R}_{\ell}$, and an inner boundary, $\partial\mathcal{R}_{\text{i}}$ as shown in Fig. 2.1. Hence, $\partial\mathcal{R} = \partial\mathcal{R}_{\text{o}} \cup \partial\mathcal{R}_{\ell} \cup \partial\mathcal{R}_{\text{i}}$, with $\partial\mathcal{R}_{\text{o}} = \{(r, \theta, \varphi) \in \mathcal{S} : r = r_{\text{eye}},\ \varphi_{\text{retina}} \leq \varphi \leq \frac{\pi}{2}\}$, $\partial\mathcal{R}_{\ell} = \{(r, \theta, \varphi) \in \mathcal{S} : r_{\text{eye}} - r_{\text{retina}} \leq r \leq 1,\ \varphi = \varphi_{\text{retina}}\}$, $\partial\mathcal{R}_{\text{i}} = \{(r, \theta, \varphi) \in \mathcal{S} : r = r_{\text{eye}} - r_{\text{retina}},\ \varphi_{\text{retina}} \leq \varphi \leq \frac{\pi}{2}\}$. The parameters $r_{\text{eye}}, r_{\text{retina}}$ and φ_{retina} are chosen to match that of humans, but can easily be chosen to study other species with similar



$\mathcal{R}$ $\partial\mathcal{R} = \partial\mathcal{R}_o \cup \partial\mathcal{R}_l \cup \partial\mathcal{R}_i$ $\mathcal{R} \subset \mathcal{S}$

Figure 2.1: Left: The retina, $\mathcal{R}$ is shown. Middle: The outer boundary, $\partial\mathcal{R}_{\mathrm{o}}$, lateral boundary, $\partial\mathcal{R}_\ell$, and inner boundary, $\partial\mathcal{R}_{\mathrm{i}}$ of the retina are shown. Right: The retina is centered across the north pole of the eye, $\mathcal{S}$. In this set up the cornea would be located around the south pole of the eye.

eye structure to humans.

2.2 Model Derivation

As shown in Fig. 2.1, we segment the eye into the sensory part, the retina $\mathcal{R}$, and the rest of the eye, $\mathcal{S} \setminus \mathcal{R}$, which will consist of various parts including the vitreous chamber and the lens, assumed to be homogeneous. We use the bi-domain/multi-domain equations to model the retina, while we model the passive region using the mono-domain equation. The interface between the two regions is the lateral boundary, $\partial\mathcal{R}_\ell$, and the inner boundary $\partial\mathcal{R}_{\mathrm{i}}$.

2.2.1 Bi-domain Equations Setup

For the sake of completeness, we present a derivation of the bi-domain equations. Many similar derivations can be found in the literature, for example [66]. Let $\phi_{\mathrm{i}}, \phi_{\mathrm{e}}$ be the intracellular and the extracellular potentials of the retina, respectively, and ϕ_{s} be the potential of the rest of the eye. Using the microscopic version of Ohm's law

we get

$$J_{\mathrm{i}} = \mu_{\mathrm{i}} E_{\mathrm{i}} = -\mu_{\mathrm{i}} \nabla \phi_{\mathrm{i}},$$
$$J_{\mathrm{e}} = \mu_{\mathrm{e}} E_{\mathrm{e}} = -\mu_{\mathrm{e}} \nabla \phi_{\mathrm{e}},$$
$$J_{\mathrm{s}} = \mu_{\mathrm{s}} E_{\mathrm{s}} = -\mu_{\mathrm{s}} \nabla \phi_{\mathrm{s}},$$

in which $\mu_{\mathrm{i}}, \mu_{\mathrm{e}}$ are the conductivity of the intracellular and extracellular domains of the retina, and μ_{s} is the conductivity of the interior of the eye; $J_{\mathrm{i}}, J_{\mathrm{e}}$ denote the intracellular and extracellular current densities in the retina, and J_{s} the current density in the rest of the eye; and $E_{\mathrm{i}}, E_{\mathrm{e}}$ denote the intracellular and extracellular electric fields in the retina, and E_{s} the electric field in the rest of the eye. In our setting, the conductivities will be (symmetric positive definite) tensors that are functions of space as discussed in Section 4.1.

As is typical in electrostatics, our assumption that charge cannot accumulate at any point on the passive region of the eye takes the form

$$\nabla \cdot J_{\mathrm{s}} = 0.$$

As each point in the retina resides in both the intracellular and extracellular domain, our assumption about charge accumulation takes the form

$$\nabla \cdot (J_{\mathrm{e}} + J_{\mathrm{i}}) = 0.$$

In each domain, transmembrane currents, capacitive currents, and any applied currents appear as sources

$$\nabla \cdot J_{\mathrm{i}} = -\nabla \cdot (\mu_{\mathrm{i}} \nabla \phi_{\mathrm{i}}) = -\frac{1}{\chi}(C_{\mathrm{m}} \frac{\partial V_{\mathrm{m}}}{\partial t} + I_{\mathrm{m}}),$$
$$\nabla \cdot J_{\mathrm{e}} = -\nabla \cdot (\mu_{\mathrm{e}} \nabla \phi_{\mathrm{e}}) = \frac{1}{\chi}(C_{\mathrm{m}} \frac{\partial V_{\mathrm{m}}}{\partial t} + I_{\mathrm{m}}),$$

in which χ is the volume-to-surface ratio of the cell membrane, C_{m} is the cell membrane capacitance, I_{m} is the transmembrane current density, and $V_{\mathrm{m}} = \phi_{\mathrm{i}} - \phi_{\mathrm{e}}$ is the membrane voltage, the difference between the intracellular and extracellular potential. The sign difference between the right hand sides is expected since current exiting the intracellular region enters the extracellular region.

We choose the transmembrane currents to be given by a conductance-based, rod-photoreceptor model [64], the Kamiyama model. However, our model is adaptable to any choice of transmembrane currents. The flexibility of the model, both geometrically (see Section 2.1) and physiologically, makes it readily adaptable to study numerous other species. In transmembrane current models where the total current is reported rather than the density, one needs to divide by the surface area of the cell membrane, which may be absorbed into χ. Parameters associated with the transmembrane currents are permitted to depend on space if required. Since cone and rod photoreceptors have similar ionic currents in their inner segments [10], we can also model cones by reducing some time constants in the Kamiyama model. As photoreceptors are the main retinal cells of our interest, we used the same set of equations for the transmembrane currents throughout the entire retina.

To address boundary conditions of this system, we start by assuming that the eye is surrounded by perfectly insulating material (so that the current cannot leave the eye)

$$\begin{aligned} n_{\mathrm{o}} \cdot (\mu_{\mathrm{i}} \nabla \phi_{\mathrm{i}}) &= 0 && (\text{on } \partial\mathcal{R}_{\mathrm{o}}), \\ n_{\mathrm{o}} \cdot (\mu_{\mathrm{e}} \nabla \phi_{\mathrm{e}}) &= 0 && (\text{on } \partial\mathcal{R}_{\mathrm{o}}), \\ n_{\mathrm{s}} \cdot (\mu_{\mathrm{s}} \nabla \phi_{\mathrm{s}}) &= 0 && (\text{on } \partial\mathcal{S} \setminus \partial\mathcal{R}_{\mathrm{o}}), \end{aligned}$$

in which n_{s} is the normal vector to $\partial\mathcal{S} = \{(r, \theta, \varphi) \in \mathcal{S} : r = r_{\mathrm{eye}}\}$ and n_{o} is the normal to $\partial\mathcal{R}_{\mathrm{o}}$. On the boundary of the retina and the rest of the eye, where the

transition of a bi-domain to a mono-domain occurs, we require that the extracellular potential and the current be continuous

$$\begin{aligned} \phi_{\mathrm{e}} &= \phi_{\mathrm{s}} && (\text{on } \partial\mathcal{R}_{\mathrm{i}} \cup \partial\mathcal{R}_{\ell}), \\ n_{\mathrm{x}} \cdot (\mu_{\mathrm{i}} \nabla\phi_{\mathrm{i}} + \mu_{\mathrm{e}} \nabla\phi_{\mathrm{e}}) &= n_{\mathrm{x}} \cdot (\mu_{\mathrm{s}} \nabla\phi_{\mathrm{s}}) && (\text{on } \partial\mathcal{R}_{\mathrm{x}} \text{ for } \mathrm{x} = \mathrm{i}, \ell). \end{aligned} \tag{2.1}$$

Following Tung [120], we make the additional assumption that the intracellular domain of the retina is isolated from the rest of the eye, which gives

$$\begin{aligned} n_{\mathrm{x}} \cdot (\mu_{\mathrm{i}} \nabla\phi_{\mathrm{i}}) &= 0 && (\text{on } \partial\mathcal{R}_{\mathrm{x}} \text{ for } \mathrm{x} = \mathrm{i}, \ell), \\ n_{\mathrm{x}} \cdot (\mu_{\mathrm{e}} \nabla\phi_{\mathrm{e}}) &= n_{\mathrm{x}} \cdot (\mu_{\mathrm{s}} \nabla\phi_{\mathrm{s}}) && (\text{on } \partial\mathcal{R}_{\mathrm{x}} \text{ for } \mathrm{x} = \mathrm{i}, \ell). \end{aligned} \tag{2.2}$$

In summary, our model is a system of partial differential equations for unknowns $\phi_{\mathrm{i}}, \phi_{\mathrm{e}}$ (defined on $\mathcal{R}$), and ϕ_{s} (defined $\overline{\mathcal{S} \setminus \mathcal{R}}$),

$$\text{retina} \begin{cases} \nabla \cdot (\mu_{\mathrm{i}} \nabla\phi_{\mathrm{i}} + \mu_{\mathrm{e}} \nabla\phi_{\mathrm{e}}) = 0 & (\text{on } \mathcal{R}), \\ \nabla \cdot (\mu_{\mathrm{e}} \nabla\phi_{\mathrm{e}}) = -\frac{1}{\chi}\big(C_{\mathrm{m}} \frac{\partial V_{\mathrm{m}}}{\partial t} + I_{\mathrm{m}}(V_{\mathrm{m}}, \boldsymbol{X})\big) & (\text{on } \mathcal{R}), \\ V_{\mathrm{m}} = \phi_{\mathrm{i}} - \phi_{\mathrm{e}} & (\text{on } \mathcal{R}), \\ \frac{\partial \boldsymbol{X}}{\partial t} = \boldsymbol{G}(V_{\mathrm{m}}, \boldsymbol{X}) & (\text{on } \mathcal{R}), \end{cases}$$

$$\text{retina boundary} \begin{cases} n_{\mathrm{x}} \cdot (\mu_{\mathrm{i}} \nabla\phi_{\mathrm{i}}) = 0 & (\text{on } \partial\mathcal{R}_{\mathrm{x}} \text{ for } \mathrm{x} = \mathrm{o}, \mathrm{i}, \ell), \\ n_{\mathrm{o}} \cdot (\mu_{\mathrm{e}} \nabla\phi_{\mathrm{e}}) = 0 & (\text{on } \partial\mathcal{R}_{\mathrm{o}}), \\ \phi_{\mathrm{e}} = \phi_{\mathrm{s}} & (\text{on } \partial\mathcal{R}_{\mathrm{i}} \cup \partial\mathcal{R}_{\ell}), \\ n_{\mathrm{x}} \cdot (\mu_{\mathrm{e}} \nabla\phi_{\mathrm{e}}) = n_{\mathrm{x}} \cdot (\mu_{\mathrm{s}} \nabla\phi_{\mathrm{s}}) & (\text{on } \partial\mathcal{R}_{\mathrm{x}} \text{ for } \mathrm{x} = \mathrm{i}, \ell), \end{cases}$$

$$\text{rest of the eye} \begin{cases} \nabla \cdot (\mu_{\mathrm{s}} \nabla\phi_{\mathrm{s}}) = 0 & (\text{on } \mathcal{S} \setminus \mathcal{R}), \\ n_{\mathrm{s}} \cdot (\mu_{\mathrm{s}} \nabla\phi_{\mathrm{s}}) = 0 & (\text{on } \partial\mathcal{S} \setminus \partial\mathcal{R}_{\mathrm{o}}), \end{cases}$$

in which t is time, n_{s} is the outward normal vector to $\partial\mathcal{S}$, n_{x} is the normal to $\partial\mathcal{R}_{\mathrm{x}}$ (pointing away from the retina) for $\mathrm{x} = \mathrm{o}, \mathrm{i}, \ell$, $\boldsymbol{X}$ contains all of the auxiliary dynamic

variables from the transmembrane currents model (including channel gating variables and concentrations of proteins involved in phototransduction), and $\boldsymbol{G}$ describes their evolution in time.

We eliminate ϕ_{i} and re-arrange for a dynamic equation for V_{m},

$$\nabla \cdot (\mu_{\mathrm{s}} \nabla \phi_{\mathrm{s}}) = 0 \qquad (\text{on } \mathcal{S} \setminus \mathcal{R}), \tag{2.3}$$

$$\nabla \cdot (\mu_{\mathrm{i}} \nabla (V_{\mathrm{m}} + \phi_{\mathrm{e}}) + \mu_{\mathrm{e}} \nabla \phi_{\mathrm{e}}) = 0 \qquad (\text{on } \mathcal{R}), \tag{2.4}$$

$$\frac{\partial V_{\mathrm{m}}}{\partial t} = -\frac{1}{C_{\mathrm{m}}} (\chi \nabla \cdot (\mu_{\mathrm{e}} \nabla \phi_{\mathrm{e}}) + I_{\mathrm{m}}(V_{\mathrm{m}}, \boldsymbol{X})) \qquad (\text{on } \mathcal{R}), \tag{2.5}$$

$$\frac{\partial \boldsymbol{X}}{\partial t} = \boldsymbol{G}(V_{\mathrm{m}}, \boldsymbol{X}) \qquad (\text{on } \mathcal{R}). \tag{2.6}$$

Equations (2.3) and (2.4) are the elliptic constraints on the evolution of the system in time. In Section 4.1, we see explicitly that the discretization of this PDE system is a system of differential algebraic equations (DAE). The elliptic constraints become the algebraic restriction on the system.

2.2.2 Uniqueness and Boundary Conditions

By inspecting the PDE system (2.3) to (2.6), it is clear that its solutions are not unique. That is, if $V_{\mathrm{m}}, \phi_{\mathrm{e}}, \phi_{\mathrm{s}}$ are solutions then so are $V_{\mathrm{m}}, \phi_{\mathrm{e}} + c, \phi_{\mathrm{s}} + c$ for any constant c. Potentials are not unique so long as the potential difference is unchanged. Since we will be solving this system in time as well, it is important to note that c is allowed to depend on time. So fixing the time dependent constant c, makes the PDE system (2.3) to (2.6) well-posed.

We choose c to ensure that

$$\int_{\partial \mathcal{S} \setminus \partial \mathcal{R}} \phi_{\mathrm{s}} + \int_{\partial \mathcal{R}_{\mathrm{o}}} \phi_{\mathrm{e}} = 0 \tag{2.7}$$

at all times. The motivation behind this choice can be thought of as selecting the

ground electrical potential to be zero. Let ϕ_{o} be the harmonic function on $\mathbb{R}^3 \backslash \mathcal{S}$ whose values on $\partial\mathcal{S}$ matches ϕ_{s} and ϕ_{e}, and let $\phi_\infty = \lim_{r\to\infty} \phi_{\mathrm{o}}$. Hence, ϕ_{o} is an extension of ϕ_{e} (on $\partial\mathcal{R}_{\mathrm{o}}$) and ϕ_{s} (on $\partial\mathcal{S} \setminus \partial\mathcal{R}$), while ϕ_∞ can be thought of as the ground potential at infinity. Using the mean value property on the Kelvin transform of ϕ_{o} shows that our assumption is tantamount to $\phi_\infty = \int_{\partial\mathcal{S}} \phi_{\mathrm{o}} = 0$.

2.2.3 Multi-domain Equations Setup

We will derive the multi-domain equations with an arbitrary number of distinct intracellular domains, say there are q of them with intracellular potentials $\phi_{\mathrm{i}}^1, \phi_{\mathrm{i}}^2, \ldots, \phi_{\mathrm{i}}^q$. The derivation will be very similar to that of the bi-domain equations (Section 2.2.1), so many of the details will be omitted. We also omit redefining variables introduced previously.

Considering the multi-domain setup, assuming that charge cannot accumulate at any point gives

$$\nabla \cdot \left(J_{\mathrm{e}} + \sum_{j=1}^{q} J_{\mathrm{i}}^j \right) = 0,$$

in which J_{i}^j is the current density of the jth intracellular domain. Different photoreceptors are electrically coupled via channels called gap junctions [35, 98, 72]. Hence, gap junctional currents also appear as a source of current

$$\nabla \cdot J_{\mathrm{i}}^j = -\nabla \cdot (\mu_{\mathrm{i}}^j \nabla \phi_{\mathrm{i}}^j) = -\frac{1}{\chi^j} \left(C_{\mathrm{m}}^j \frac{\partial V_{\mathrm{m}}^j}{\partial t} + I_{\mathrm{m}}^j + \sum_{k=1}^{q} g_{jk} (\phi_{\mathrm{i}}^j - \phi_{\mathrm{i}}^k) \right),$$

in which μ_{i}^j is the conductivity of the jth intracellular domain, χ^j is the volume-to-surface ratio of the cell membrane of the jth cell-type, C_{m}^j is the cell membrane capacitance of the jth cell-type, I_{m}^j is the transmembrane current density of the jth cell-type, $V_{\mathrm{m}}^j = \phi_{\mathrm{i}}^j - \phi_{\mathrm{e}}$ is the difference of the jth intracellular domain potential and

extracellular potential, and g_{jk} is the gap junctional conductance between the jth and kth intracellular domains (we set $g_{jj} = 0$, for $j = 1, \ldots, q$).

The boundary conditions are handled similarly to the bi-domain equations. The overall multi-domain model will also be a system of partial differential equations for unknowns $\phi_{\mathrm{i}}^j, \phi_{\mathrm{e}}$, and ϕ_{s} (for $j = 1, \ldots, q$). These are

$$\text{retina}\begin{cases} \nabla \cdot (\mu_{\mathrm{e}} \nabla \phi_{\mathrm{e}} + \sum_{j=1}^{q} \mu_{\mathrm{i}}^j \nabla \phi_{\mathrm{i}}^j) = 0 & (\text{on } \mathcal{R}), \\ \nabla \cdot (\mu_{\mathrm{i}}^j \nabla \phi_{\mathrm{i}}^j) = \frac{1}{\chi^j}\left(C_{\mathrm{m}}^j \frac{\partial V_{\mathrm{m}}^j}{\partial t} + I_{\mathrm{m}}^j + \sum_{k=1}^{q} g_{jk}(\phi_{\mathrm{i}}^j - \phi_{\mathrm{i}}^k) \right) & (\text{on } \mathcal{R},\ j = 1, \ldots, q), \\ V_{\mathrm{m}}^j = \phi_{\mathrm{i}}^j - \phi_{\mathrm{e}} & (\text{on } \mathcal{R},\ j = 1, \ldots, q), \\ \frac{\partial \boldsymbol{X}^j}{\partial t} = \boldsymbol{G}^j(V_{\mathrm{m}}^j, \boldsymbol{X}^j) & (\text{on } \mathcal{R},\ j = 1, \ldots, q), \end{cases}$$

$$\text{retina boundary}\begin{cases} n_{\mathrm{x}} \cdot (\mu_{\mathrm{i}}^j \nabla \phi_{\mathrm{i}}^j) = 0 & (\text{on } \partial\mathcal{R}_{\mathrm{x}} \text{ for } \mathrm{x} = \mathrm{o}, \mathrm{i}, \ell;\ j = 1, \ldots, q), \\ n_{\mathrm{o}} \cdot (\mu_{\mathrm{e}} \nabla \phi_{\mathrm{e}}) = 0 & (\text{on } \partial\mathcal{R}_{\mathrm{o}}), \\ \phi_{\mathrm{e}} = \phi_{\mathrm{s}} & (\text{on } \partial\mathcal{R}_{\mathrm{i}} \cup \partial\mathcal{R}_{\ell}), \\ n_{\mathrm{x}} \cdot (\mu_{\mathrm{e}} \nabla \phi_{\mathrm{e}}) = n_{\mathrm{x}} \cdot (\mu_{\mathrm{s}} \nabla \phi_{\mathrm{s}}) & (\text{on } \partial\mathcal{R}_{\mathrm{x}} \text{ for } \mathrm{x} = \mathrm{i}, \ell), \end{cases}$$

$$\text{rest of the eye}\begin{cases} \nabla \cdot (\mu_{\mathrm{s}} \nabla \phi_{\mathrm{s}}) = 0 & (\text{on } \mathcal{S} \setminus \mathcal{R}), \\ n_{\mathrm{s}} \cdot (\mu_{\mathrm{s}} \nabla \phi_{\mathrm{s}}) = 0 & (\text{on } \partial\mathcal{S} \setminus \partial\mathcal{R}_{\mathrm{o}}), \end{cases}$$

in which $\boldsymbol{X}^j$ contains all of the auxiliary dynamic variables from the transmembrane currents model for the jth cell type and $\boldsymbol{G}^j$ describes their evolution in time.

As before, we obtain the following PDE system for the multi-domain equations:

$$\nabla \cdot (\mu_{\mathrm{s}} \nabla \phi_{\mathrm{s}}) = 0 \qquad (\text{on } \mathcal{S} \setminus \mathcal{R}), \tag{2.8}$$

$$\nabla \cdot \left(\mu_{\mathrm{e}} \nabla \phi_{\mathrm{e}} + \sum_{j=1}^{q} \mu_{\mathrm{i}}^j \nabla (V_{\mathrm{m}}^j + \phi_{\mathrm{e}}) \right) = 0 \qquad (\text{on } \mathcal{R}), \tag{2.9}$$

$$\begin{aligned} \frac{\partial V_{\mathrm{m}}^j}{\partial t} = \frac{1}{C_{\mathrm{m}}^j} \Bigg(& \chi^j \nabla \cdot \left(\mu_{\mathrm{i}}^j \nabla (V_{\mathrm{m}}^j + \phi_{\mathrm{e}}) \right) \\ & - I_{\mathrm{m}}^j(V_{\mathrm{m}}, \boldsymbol{X}^j) - \sum_{k=1}^{q} g_{jk}(V_{\mathrm{m}}^j - V_{\mathrm{m}}^k) \Bigg) \qquad (\text{on } \mathcal{R},\ j = 1, \ldots, q), \end{aligned} \tag{2.10}$$

$$\frac{\partial \boldsymbol{X}^j}{\partial t} = \boldsymbol{G}^j(V_{\mathrm{m}}^j, \boldsymbol{X}^j) \qquad (\text{on } \mathcal{R},\ j = 1, \ldots, q). \tag{2.11}$$

Chapter 3

Iterative Methods for Solving Linear Systems

Given an $n \times n$ nonsingular matrix, A, solving the linear system

$$Ax = b, \tag{3.1}$$

continues to be a main subject in numerical research up to this point. Beginning in the 1970s, there has been a shift towards solving large linear systems iteratively as opposed to directly. This is because direct solvers are prohibitively slow for very large n. Indeed, in the same time span that computer hardware had a speedup factor of 10^9, our ability to solve large systems improved only by a factor of 10^3 [119]. A main advantage of iterative methods is the ability to stop when a suitable, but not necessarily exact, solution to (3.1) is found. This can significantly cut down the computational costs.

In this chapter, we introduce, and provide background on, the iterative methods that we use in solving linear systems that arise from using Newton's method on the discretized PDE system (see Chapter 4). We conclude this chapter with a discussion on preconditioners, especially as it pertains to iterative methods. A reader who is

mainly concerned with how we simulated our model rather than the details on the techniques used can safely skip this chapter.

3.1 Fixed-point Iteration

The equation

$$x_{n+1} = f(x_n)$$

is called a functional iteration [67]. If we assume that f is a continuous function and $x_n \to x$ is a convergent sequence, it is clear that $x = f(x)$. In other words, the limit of the sequence generated by the functional iteration is a fixed-point of the function f.

It is often the case that a mathematical problem can be framed as a fixed-point iteration of some operator. An example of this is solving the linear system (3.1) using the Jacobi iteration. If we let D be the diagonal matrix with the same diagonal as A, assuming D is invertible, then the Jacobi iteration is the fixed-point iteration defined by

$$Dx_{n+1} = b - (A - D)x_n.$$

Indeed, if $x_n \to x$, then $Dx = b - (A - D)x$ is equivalent to x being a solution to (3.1).

In fact, given any splitting $A = B + (A - B)$, solving the linear system (3.1) can be casted as a fixed point iteration, namely

$$Bx_{n+1} = b - (A - B)x_n. \tag{3.2}$$

Furthermore, it can be shown that the above iteration will always converge if the spectral radius of the matrix $B^{-1}(A - B)$ is strictly less than unity [133, 104].

Equation (3.2) is also a semi-explicit iteration for solving (3.1). This can be seen by writing (3.1) as

$$Bx + (A - B)x = b$$

and solving the B part implicitly and the $A - B$ part explicitly.

3.1.1 Richardson-D'Jakonov Iteration

In 1911, Richardson [97] proposed solving the linear equation $Ax = 0$ iteratively using

$$x_{j+1} = x_j - \gamma_j A x_j,$$

in which x_j is a sequence that converges to the solution of the linear equation and the iteration parameters $\gamma_j \in \mathbb{R}$ are allowed to vary with each iteration. The Richardson iteration can be viewed as a fixed-point iteration based on the splitting $\gamma_j A = I + (\gamma_j A - I)$ [104]. Around 50 years later, D'Jakonov [32, 33] proposed a more general iteration for solving the linear equation $Ax = b$, which is given by

$$Bx_{j+1} = \gamma_j b - (\gamma_j A - B)x_j. \tag{3.3}$$

This iteration later became known as the Richardson-D'Jakonov iteration [113].

3.2 Krylov Subspace Methods

In this section, we survey the Krylov subspace methods, which are a class of iterative techniques for solving the linear system (3.1). From these methods, we only use the

generalized minimal residual (GMRES) method in our simulation (see Section 4.3.2), however, we provide this in depth introduction to place this method in context. The majority of the facts in this section can be found in modern textbooks, for example the great books of Saad [104], and Trefethen and Bau [119], upon which we relied heavily. However, we provide this survey from a historical lens, starting with the oldest method and moving on to more recent ones, while addressing, in detail, the more modern view that is found in textbooks today. Also, some relations are proved in a novel way, namely the relation between Lanczos methods and the conjugate gradient method (see Section 3.2.6).

3.2.1 Introduction to Krylov Subspace Methods

Given an initial approximation, x_0, to the solution of (3.1), a Petrov-Galerkin method seeks to find an approximation, $\tilde{x}$, to the solution of (3.1), in an affine space $x_0 + \mathcal{K}$, for some specified subspace $\mathcal{K}$ [104]. This approximation is chosen so that the residual vector $b - A\tilde{x}$ is orthogonal to a subspace $\mathcal{L}$. In this context, Petrov-Galerkin methods are synonymous to projection methods [104]. In the special case when $\mathcal{L} = \mathcal{K}$, they are usually referred to as Galerkin methods (i.e. orthogonal projection methods) [104]. A Krylov subspace method is a Petrov-Galerkin method onto the Krylov subspace

$$\mathcal{K}_m(A, r_0) = \text{span}\big\{r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0\big\},$$

in which $r_0 = b - Ax_0$ is the initial residual vector. We will simply use $\mathcal{K}_m$ to refer to the m^{th} Krylov subspace when both A and r_0 are understood from context and $\mathcal{K}_m(r_0)$ when only A is understood from context. We will only consider methods in which $\mathcal{L} = \mathcal{K}_m$ or $\mathcal{L} = A\mathcal{K}_m$.

3.2.2 Conjugate Gradient Method

The conjugate gradient method is the original Krylov subspace method (in 1951-1952), yet it is still one of the best known iterative methods for solving a sparse symmetric positive definite linear system. It was independently developed by Hestenes (with help from Rosser, Forsythe, and Paige) and by Stiefel [57, 56]. Some also credit Lanczos for independently discovering this method [74, 75, 104]. This method can easily be implemented and is found in many textbooks (e.g. [104, 119]). It is stated below.

Algorithm 1 Conjugate Gradient Method

Require: x_0 initial approximation for solution of (3.1), A symmetric positive definite

Define $r_0 = b - Ax_0,\ p_0 = r_0$

for $j = 0, 1, \ldots$ (until convergence) **do**

$$\gamma_j = \frac{\langle r_j, r_j \rangle}{\langle Ap_j, p_j \rangle}$$

$x_{j+1} = x_j + \gamma_j p_j$ % new approximation

$r_{j+1} = r_j - \gamma_j A p_j$ % new residual

$$\xi_j = \frac{\langle r_{j+1}, r_{j+1} \rangle}{\langle r_j, r_j \rangle}$$

$p_{j+1} = r_{j+1} + \xi_j p_j$ % new search direction

end for

The name of the method comes from the fact that all the search directions p_j's are A-conjugate (i.e. $\langle Ap_i, p_j \rangle = 0$ for $i \neq j$). It probably is difficult to see how Algorithm 1 solves (3.1). We will purposefully avoid directly commenting on that here. Instead we will explain how the Lanczos methods (see Section 3.2.3) work and then show that the conjugate gradient is a Lanczos method (Section 3.2.6).

3.2.3 (Symmetric) Lanczos Methods

The original Lanczos method was developed for solving the eigenvalue problem of linear operators [74]. In [74], Lanczos points that his proposed method can be used to solve linear systems, which he later demonstrated in detail in [75]. We will only

be concerned with symmetric Lanczos methods (i.e. A is symmetric in (3.1)), and, from here on, we will be referring to them simply as Lanczos methods. At the heart of Lanczos methods is the Lanczos algorithm, which, in ideal conditions, gives an orthonormal basis $\{v_1, \ldots, v_k\}$ of the Krylov subspace $\mathcal{K}_k$ (with $v_1 = \frac{r_0}{\|r_0\|}$) for $k = 1, \ldots, m$. This algorithm can be stated independently from the projection problem of interest and can also be found in many textbooks (e.g. [104, 119]). We state it below. At a first encounter, it probably is difficult to believe that the Lanczos algorithm

Algorithm 2 Lanczos Algorithm

Require: arbitrary unit vector v_1, A symmetric
  Define $\beta_1, v_0 = 0$
  for $j = 1, 2, \ldots, m$ **do**
    $\alpha_j = \langle Av_j, v_j \rangle$
    $\omega_j = Av_j - \beta_j v_{j-1} - \alpha_j v_j$ % projection onto $(\text{span}\{v_{j-1}, v_j\})^{\perp}$
    $\beta_{j+1} = \|\omega_j\|$
    if $\beta_{j+1} = 0$ **then**
      Stop
    end if
    $v_{j+1} = \frac{1}{\beta_{j+1}} \omega_j$ % normalizing
  end for

results in orthogonal vectors; we discuss this in detail in Section 3.2.5.

The ideal conditions referred to earlier are tantamount to $\dim \mathcal{K}_m = m$. This ensures that the Lanczos algorithm does not come to a premature halt (in the 'if' statement). These premature halts are referred to as breakdowns.

The other component to Lanczos methods is finding the approximation, $\tilde{x}$, of the desired solution to (3.1). In the case of Galerkin methods onto the Krylov subspace $\mathcal{K}_m$, one simple way to do this is, subsequent to the Lanczos algorithm, solve the system

$$\begin{cases} \tilde{x} = x_0 + V_m \hat{x} \\ V_m^T (b - A\tilde{x}) = 0 \end{cases},$$

in which V_m is the $n \times m$ matrix whose columns are obtained from the Lanczos algorithm (as they form a spanning set of $\mathcal{K}_m$). The orthogonality condition can be simplified further since $b - A\tilde{x} = r_0 - AV_m\hat{x} = \|r_0\|v_1 - AV_m\hat{x}$. Thus, the approximation can be found using

$$\begin{cases} \tilde{x} = x_0 + V_m\hat{x} \\ \hat{x} = \|r_0\|T_m^{-1}e_1 \end{cases}, \tag{3.4}$$

in which $T_m = V_m^T AV_m$ is an $m \times m$ matrix. In fact, it will prove useful to similarly define V_k, $T_k = V_k^T AV_k$ by taking the first k vectors from the Lanczos algorithm (Algorithm 2) for all $k = 1, \ldots, m$.

3.2.4 Arnoldi's Method

Arnoldi gave his generalization of the (symmetric) Lanczos method to the nonsymmetric case shortly after it was first proposed [8]. Much like the Lanczos method, Arnoldi's method can be broken down into: 1) finding an orthonormal basis of the Kyrlov subspace and 2) approximating the solution of (3.1) using that basis. The former is commonly known as Arnoldi's algorithm and can be found in many textbooks (e.g. [104, 119]). We state it below.

It is easily observed that Arnoldi's algorithm is theoretically equivalent to the Gram-Schmidt process and, hence, we conclude that the resulting $v_1, \ldots, v_k$ form an orthonormal basis for the Krylov subspace $\mathcal{K}_k$ for $k = 1, \ldots, m$ (again assuming no breakdown occurs).

Much like the Gram-Schmidt process can be viewed as a way to factor a matrix, with linearly independent columns, into QR form, Arnoldi's algorithm factors, under the previously mentioned ideal conditions, a matrix into its Hessenberg form. Indeed, if V_k is the $n \times k$ matrix whose columns are the output of Arnoldi's algorithm, then

Algorithm 3 Arnoldi's Algorithm

Require: arbitrary unit vector v_1
  for $j = 1, 2, \ldots, m$ **do**
    for $i = 1, 2, \ldots, j$ **do**
      $h_{ij} = \langle Av_j, v_i \rangle$
    end for
    $\omega_j = Av_j - \sum_{i=1}^{j} h_{ij} v_i$ % projection onto $(\text{span}\{v_1, \ldots, v_j\})^{\perp}$
    $h_{(j+1)j} = \|\omega_j\|$
    if $h_{(j+1)j} = 0$ **then**
      Stop
    end if
    $v_{j+1} = \frac{1}{h_{(j+1)j}} \omega_j$ % normalizing
  end for

$V_k^T A V_k$ is a Hessenberg matrix (i.e. all entries below the first subdiagonal are zeros) for all $k = 1, \ldots, m$. To show this, we define H_k to be the $k \times k$ Hessenberg matrix whose nonzero entries are defined in the first k iterations of the main loop of Algorithm 3 (disregarding $h_{(k+1)k}$). From the algorithm, one can readily observe that

$$AV_k = V_k H_k + \omega_k e_k^T \qquad (\text{for } k = 1, \ldots, m), \tag{3.5}$$

in which ω_k is the one defined in the main loop of Algorithm 3 (again, we assume we are in the case of no breakdowns). The desired result is obtained using the orthonormality of $v_1, \ldots, v_{k+1}$.

The second component of Arnoldi's method, namely finding the approximation to the solution of (3.1), can also be solved using various methods. One such method, known as the full orthogonalization method, uses the same approach discussed previously for the Lanczos method [104]. We state it below. A second approach to finding the approximation $\tilde{x}$ is introduced when we discuss the generalized minimal residual method (see Section 3.2.7).

Algorithm 4 Full Orthogonalization Method

Require: arbitrary unit vector v_1
  for $j = 1, 2, \ldots, m$ **do**
    for $i = 1, 2, \ldots, j$ **do**
      $h_{ij} = \langle Av_j, v_i \rangle$
    end for
    $\omega_j = Av_j - \sum_{i=1}^{j} h_{ij} v_i$ % projection onto $(\text{span}\{v_1, \ldots, v_j\})^{\perp}$
    $h_{(j+1)j} = \|\omega_j\|$
    if $h_{(j+1)j} = 0$ **then**
      $m = j$
      Break
    end if
    $v_{j+1} = \frac{1}{h_{(j+1)j}} \omega_j$ % normalizing
  end for
  $\hat{x} = \|r_0\| H_m^{-1} e_1$ % $r = b - A\tilde{x} \in \mathcal{L}^{\perp} = \mathcal{K}_m^{\perp}$
  $\tilde{x} = x_0 + V_m \hat{x}$ % $\tilde{x} \in x_0 + \mathcal{K}_m$

3.2.5 Lanczos Methods are a Special Case of Arnoldi's Method

Now that it is clear that the columns of V_k, obtained from Arnoldi's algorithm, are an othonormal basis of $\mathcal{K}_k$, the next proposition is the key to showing the relation between Arnoldi's algorithm and the Lanczos algorithm.

Proposition 3.1. *If A is a symmetric $n \times n$ matrix, then the Hessenberg matrix H_k, defined using Arnoldi's algorithm, is a tridiagonal matrix (for all $k = 1, \ldots, m$).*

Proof. For any $k = 1, \ldots, m$, as $H_k = V_k^T A V_k$ is a symmetric matrix it suffices to show $(H_k)_{ij} = 0$ for $i > j + 1$. So we need to show $\langle v_i, Av_j \rangle = 0$. Since $v_j \in \mathcal{K}_j$, we have $Av_j \in \mathcal{K}_{j+1} = \text{span}\{v_1, \ldots, v_{j+1}\}$, which gives the desired result by the orthogonality of the v_i's. □

So, in the case of A being symmetric, we have

$$H_k = \begin{bmatrix} h_{11} & h_{12} & \cdot & \cdot & \cdot & h_{1k} \\ h_{21} & h_{22} & \cdot & \cdot & \cdot & h_{2k} \\ & \cdot & \cdot & & & \cdot \\ & & \cdot & \cdot & & \cdot \\ 0 & & & \cdot & \cdot & \cdot \\ & & & h_{k(k-1)} & h_{kk} & \end{bmatrix} = \begin{bmatrix} h_{11} & h_{21} & & & & \\ h_{21} & h_{22} & \cdot & & 0 & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & 0 & & \cdot & \cdot & h_{k(k-1)} \\ & & & h_{k(k-1)} & h_{kk} & \end{bmatrix}.$$

Noting the shape of H_k, it is not too difficult to see that the j^{th} projection in Arnoldi's algorithm is, theoretically, equivalent to projecting onto $(\text{span}\{v_{j-1}, v_j\})^{\perp}$. Since $h_{jj} = \alpha_j$ and $h_{(j+1),j} = \beta_{j+1}$, we get that the Lanczos methods are a special case of Arnoldi's method since the algorithms are identical after simplifying to the symmetric case. Thus, we see that the Lanczos algorithm indeed generates an orthonormal basis and in that case

$$T_k = H_k = \begin{bmatrix} \alpha_1 & \beta_1 & & & & \\ \beta_1 & \alpha_2 & \cdot & & 0 & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & 0 & & \cdot & \cdot & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_k & \end{bmatrix}.$$

3.2.6 Conjugate Gradient Method is a Lanczos Method

The goal of the Lanczos (and Arnoldi's) algorithm is to form a basis of $\mathcal{K}_m$ by iteratively enlarging a basis of $\mathcal{K}_k$ for $k = 1, \ldots, m$ (in that order). So we can potentially reduce the cost of Lanczos methods (and Arnoldi's) by finding approximations $\tilde{x}_k \in x_0 + \mathcal{K}_k$ as follows.

Algorithm 5 is a poor implementation since, other than not recomputing v_i's, the approximations are computed independently. We now show that, in the case of A being positive definite, we can pose the above Lanczos method as an iterative method that avoids these high costs. In fact, the iterative method we derive is the conjugate gradient method, thus showing that the conjugate gradient method is a Lanczos method

Algorithm 5 Lanczos Method

Require: x_0 initial approximation for solution of (3.1)

Define $\beta_1, v_0 = 0, r_0 = b - Ax_0, v_1 = \frac{1}{\|r_0\|} r_0$

for $k = 1, 2, \ldots, m$ **do**

$\hat{x}_k = \|r_0\| T_k^{-1} e_1$

$\tilde{x}_k = x_0 + V_k \hat{x}_k$ % k^{th} approximation in the affine subspace $x_0 + \mathcal{K}_k$

if x_k is a satisfactory approximation **then**

Stop

end if

$\alpha_k = \langle Av_k, v_k \rangle$

$\omega_k = Av_k - \beta_k v_{k-1} - \alpha_k v_k$ % projection onto $(\text{span}\{v_{k-1}, v_k\})^{\perp}$

$\beta_{k+1} = \|\omega_k\|$

if $\beta_{k+1} = 0$ **then**

Stop

end if

$v_{k+1} = \dfrac{1}{\beta_{k+1}} \omega_k$ % normalizing

end for

in the special case of symmetric positive definite matrices.

As A is a symmetric positive definite matrix, so is $T_k = V_k^T A V_k$ for all $k = 1, \ldots, m$, and so it has a Cholesky factorization. In fact, we can write $T_k = L_k D_k L_k^T$, for some unique L_k unit lower triangular (diagonal entries are all 1) and unique D_k diagonal matrix. The particular structure of the matrices in this decomposition is the key to have the desired iterative formulation. By comparing entries we get

$$L_k = \begin{bmatrix} 1 & & & \\ \mu_1 & 1 & & 0 \\ & \mu_2 & \ddots & \\ & & \ddots & 1 \\ 0 & & & \mu_{k-1} \end{bmatrix}, \qquad D_k = \begin{bmatrix} d_1 & & & \\ & d_2 & & 0 \\ & & \ddots & \\ & & & d_k \\ 0 & & & \end{bmatrix},$$

in which $d_1 = \alpha_1, d_i = \alpha_i - \beta_{i-1}\mu_{i-1}$ for $i = 2, \ldots, k$ and $\mu_i = \dfrac{\beta_i}{d_i}$ for $i = 1, \ldots, k-1$. So the k^{th} approximation can be written as

$$\tilde{x}_k = x_0 + \left(\|r_0\| V_k L_k^{-T}\right)\left(D_k^{-1} L_k^{-1} e_1\right).$$

Now we examine the solutions of $L_k D_k y_k = e_1$. We can observe that

$$L_{k+1}D_{k+1} = \begin{bmatrix} d_1 & & & & \\ \beta_1 & d_2 & & 0 & \\ & \beta_2 & \ddots & & \\ & & \ddots & \ddots & \\ & 0 & & \beta_{k-1} & d_k \\ & & & \beta_k & d_{k+1} \end{bmatrix} = \begin{bmatrix} & & & 0 \\ & L_k D_k & & 0 \\ & & & \vdots \\ & & & \\ 0 & 0 & \cdots & \beta_k & d_{k+1} \end{bmatrix},$$

and so $y_{k+1} = \begin{bmatrix} y_k; -\dfrac{\beta_k y_k^k}{d_{k+1}} \end{bmatrix}$ can be easily obtained from y_k (note that y_i^j refers to the j^{th} entry of y_i). As the j^{th} entry of all the y_k's (if defined) are equal, we will simply write y^j to denote that value. The next lemma shows the relation between the residual vectors and the orthonormal basis obtained from Lanczos method.

Lemma 3.2. *The residual vector $r_k = b - A\tilde{x}_k$ is a constant multiple of v_{k+1} for all $k = 1, \ldots, m-1$. In particular, r_k's form an orthogonal set as well.*

Proof. Since we already know v_k's form an orthonormal set, it suffices to show that r_k is a constant multiple of v_{k+1} for all $k = 0, \ldots, m-1$. Clearly $r_0 = \|r_0\| v_1$ is a constant multiple of v_1. For $k > 0$ we have

$$\begin{aligned} r_k &= b - A\tilde{x}_k \\ &= b - Ax_0 - AV_k\hat{x}_k \\ &= r_0 - \|r_0\| V_k e_1 - \|r_0\| \omega_k e_k^T T_k^{-1} e_1 && \text{(using (3.4) and (3.5))} \\ &= r_0 - \|r_0\| V_k e_1 - \|r_0\| \omega_k \big(e_k^T L_k^{-T}\big) y_k && \text{(using the } L_k D_k L_k^T \text{ factorization of } T_k\text{)} \\ &= r_0 - \|r_0\| V_k e_1 - \|r_0\| \omega_k e_k^T y_k && (L_k \text{ is unit lower triangular}). \end{aligned}$$

Since $r_0 = \|r_0\| v_1$ and $\omega_k = \beta_k v_{k+1}$ we get

$$r_k = -\|r_0\| \beta_k y^k v_{k+1}, \tag{3.6}$$

as needed to conclude the proof. □

We now turn our attention to the solutions of $P_k' L_k^T = V_k$. Using Lemma 3.2 and the fact that the conjugate gradient method only relies on residuals rather than the v_i's, we instead focus on $-\|r_0\| P_k' L_k^T B_k = -\|r_0\| V_k B_k =: R_k$, in which $R_k = \begin{bmatrix} r_0 & r_1 & \dots & r_{k-1} \end{bmatrix}$ is an $n \times k$ matrix whose columns are the residual vectors, and

$$B_k := \begin{bmatrix} -1 & & & \\ & \beta_1 y^1 & & 0 \\ & & \ddots & \\ & & & \ddots \\ & 0 & & \\ & & & \beta_{k-1} y^{k-1} \end{bmatrix}.$$

Right multiplications by a diagonal matrix scale all columns by the corresponding diagonal entry, as opposed to left multiplications, which scale rows. Thus, it is not too difficult to see that $L_k^T B_k = B_k L_k'^T$, in which

$$L_k' := \begin{bmatrix} 1 & & & & \\ -\beta_1 y^1 \mu_1 & 1 & & & 0 \\ & \frac{\beta_2 y^2 \mu_2}{\beta_1 y^1} & \ddots & & \\ & & \ddots & \ddots & \\ & & & \ddots & 1 \\ 0 & & & \frac{\beta_{k-1} y^{k-1} \mu_{k-1}}{\beta_{k-2} y^{k-2}} & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ -\mu_1^2 & 1 & & & 0 \\ & -\mu_2^2 & \ddots & & \\ & & \ddots & \ddots & \\ & & & \ddots & 1 \\ 0 & & & -\mu_{k-1}^2 & 1 \end{bmatrix}.$$

The second equality above follows from the recursive definition of y^k. Defining $P_k = -\|r_0\| P_k' B_k$, we get $P_k L_k'^T = R_k$ and if $P_k = \begin{bmatrix} p_0 & p_1 & \dots & p_{k-1} \end{bmatrix}$ we get $p_0 = r_0$ and the recursive relation

$$p_k = r_k + \mu_k^2 p_{k-1} \qquad \text{(for all } k = 1, \dots, m-1\text{).} \tag{3.7}$$

Putting it all together, we get

$$\begin{aligned}
\tilde{x}_k &= x_0 + \big(\|r_0\| V_k L_k^{-T}\big)\big(D_k^{-1} L_k^{-1} e_1\big) \\
&= x_0 - \big((-\|r_0\| V_k B_k)\big(B_k^{-1} L_k^{-T}\big)\big) y_k \\
&= x_0 - \Big(R_k \big(L_k^T B_k\big)^{-1}\Big) y_k \\
&= x_0 - \big(R_k L_k'^{-T}\big) B_k^{-1} y_k \\
&= x_0 - P_k B_k^{-1} y_k \\
&= x_0 + P_k \begin{pmatrix} \frac{1}{d_1} \\ \vdots \\ \frac{1}{d_k} \end{pmatrix} \qquad \text{(using } y^1 = \frac{1}{d_1} \text{ and the recursive definition of } y_k\text{)}
\end{aligned}$$

Since $P_k = \begin{bmatrix} P_{k-1} & p_{k-1} \end{bmatrix}$ we get the following recursive definition of $\tilde{x}_k$ and, thus, r_k

$$\tilde{x}_k = \tilde{x}_{k-1} + \frac{1}{d_k} p_{k-1} \qquad \text{(for all } k = 1, \ldots, m\text{)}, \tag{3.8}$$

$$r_k = r_{k-1} - \frac{1}{d_k} A p_{k-1} \qquad \text{(for all } k = 1, \ldots, m\text{)}. \tag{3.9}$$

Now we prove a technical lemma before proving our main result.

Lemma 3.3. *(a) The p_k vectors are A-conjugate for all $k = 1, \ldots, m-1$. That is,*

$\langle A p_i, p_j \rangle = 0$ *for* $i \neq j$.

(b) $\beta_k y^k = (-1)^{k+1} \prod_{i=1}^{k} \mu_i$ *for all* $k = 1, \ldots, m-1$.

(c) $\langle r_{k+1}, r_{k+1} \rangle = \langle r_0, r_0 \rangle \prod_{i=1}^{k+1} \mu_i^2$ *for all* $k = 0, \ldots, m-2$.

(d) $\langle A r_{k+1}, r_k \rangle = \langle A r_{k+1}, p_k \rangle$ *for all* $k = 0, \ldots, m-2$.

Proof. (a) It suffices to show that $(AP_k)^T P_k$ is a diagonal matrix (for any $k =$

$1, \ldots, m$). By using the various previously stated definitions, we get

$$\begin{aligned}
(AP_k)^T P_k &= P_k^T A^T P_k \\
&= \big(R_k L_k'^{-T}\big)^T A \big(R_k L_k'^{-T}\big) && \text{(since } A \text{ is symmetric)} \\
&= \|r_0\|^2 \big(V_k B_k L_k'^{-T}\big)^T A \big(V_k B_k L_k'^{-T}\big) \\
&= \big(B_k L_k'^{-T}\big)^T V_k^T A V_k \big(B_k L_k'^{-T}\big) \\
&= \big(L_k^{-T} B_k\big)^T T_k \big(L_k^{-T} B_k\big) && \text{(since } L_k^T B_k = B_k L_k'^T\text{)} \\
&= \|r_0\|^2 B_k^T L_k^{-1} T_k L_k^{-T} B_k \\
&= \|r_0\|^2 B_k D_k B_k,
\end{aligned}$$

which is clearly a diagonal matrix.

(b) This result is immediate as $\beta_1 y_1 = \dfrac{\beta_1}{d_1} = \mu_1$ and so, by induction, we get $\beta_k y^k = (-1)\dfrac{\beta_k}{d_k}\beta_{k-1} y^{k-1} = -\mu_k \cdot (-1)^k \displaystyle\prod_{i=1}^{k-1} \mu_i = (-1)^{k+1} \prod_{i=1}^{k} \mu_i$.

(c) Using the symmetry of A, (3.7) and (3.9), and Lemma 3.2, we get $\langle r_{k+1}, r_{k+1}\rangle = -\dfrac{1}{d_{k+1}}\langle A r_{k+1}, p_k\rangle = -\dfrac{1}{d_{k+1}}\big(\langle A r_{k+1}, r_k\rangle + \mu_k^2 \langle A r_{k+1}, p_{k-1}\rangle\big)$. The second term vanishes since, by the first part of this lemma, $r_{k+1} = p_{k+1} - \mu_{k+1}^2 p_k$ is A-conjugate to p_{k-1}. Using (3.6) and the previous part of this lemma, we get $-\dfrac{1}{d_{k+1}}\langle A r_{k+1}, r_k\rangle = \langle r_0, r_0\rangle \mu_{k+1} \displaystyle\prod_{i=1}^{k} \mu_i^2 \frac{1}{d_{k+1}} \langle A v_{k+2}, v_{k+1}\rangle = \langle r_0, r_0\rangle \prod_{i=1}^{k+1} \mu_i^2$.

(d) This result is immediate using (3.7) and the A-conjugacy of r_{k+1} and p_{k-1}.

□

Theorem 3.4. *The conjugate gradient method is a Lanczos method in the case of symmetric positive definite matrices.*

Proof. Comparing (3.7) to (3.9) with the conjugate gradient method, we see that, since x_0, p_0 are identical in both methods, it suffices to show that $d_{k+1} = \dfrac{1}{\gamma_k}$ for

$k = 0, \ldots, m-1$ and $\mu_{k+1}^2 = \xi_k$ for $k = 0, \ldots, m-2$. The range of k can be justified since $\mathcal{K}_m$ has at most dimension m.

The latter is an immediate corollary of Lemma 3.3. Indeed, $\mu_1^2 = \dfrac{\langle r_1, r_1 \rangle}{\langle r_0, r_0 \rangle}$ has been proven and so, using induction, we get $\langle r_{k+1}, r_{k+1} \rangle = \langle r_0, r_0 \rangle \mu_{k+1} \displaystyle\prod_{i=1}^{k} \mu_i^2 = \langle r_k, r_k \rangle \mu_{k+1}$ as needed.

As for the former, we can show this relation by first checking the $k = 0$ case. Clearly, $d_1 = \langle Av_1, v_1 \rangle = \dfrac{1}{\gamma_0}$ as $v_1 = \dfrac{1}{\|r_0\|} r_0$. Then, for $k = 1, \ldots, m-1$, repeatedly using (3.7) and Lemma 3.3 gives

$$
\begin{aligned}
\frac{1}{\gamma_k} &= \frac{\langle Ap_k, p_k \rangle}{\langle r_k, r_k \rangle} \\
&= \frac{\langle Ap_k, r_k \rangle}{\langle r_k, r_k \rangle} \\
&= \frac{\langle Ar_k, r_k \rangle + \mu_k^2 \langle Ap_{k-1}, r_k \rangle}{\langle r_k, r_k \rangle} \\
&= \frac{\langle Ar_k, r_k \rangle + \mu_k^2 \langle Ar_k, p_{k-1} \rangle}{\langle r_k, r_k \rangle} && \text{(since } A \text{ is symmetric)} \\
&= \frac{\langle Ar_k, r_k \rangle + \mu_k^2 \langle Ar_k, r_{k-1} \rangle}{\langle r_k, r_k \rangle} \\
&= \frac{\langle r_0, r_0 \rangle \left(\beta_k y^k\right)^2 (\langle Av_{k+1}, v_{k+1} \rangle + \mu_k \langle Av_{k+1}, v_k \rangle)}{\langle r_k, r_k \rangle} && \text{(using (3.6))} \\
&= \alpha_{k+1} + \mu_k \beta_k,
\end{aligned}
$$

as needed to conclude the proof. □

3.2.7 Generalized Minimal Residual Method

The generalized minimal residual (GMRES) method, proposed by Saad and Schultz in 1986, is one of the more recent and versatile Krylov subspace methods [105]. At its core, GMRES is an Arnoldi method and thus can be used to find solutions of (3.1), even when A is nonsymmetric and/or indefinite. GMRES is a Petrov-Galerkin

method onto $\mathcal{K} = \mathcal{K}_m$ orthogonal to $\mathcal{L} = A\mathcal{K}_m$. Consistent with our prior exposition, one way to find the approximation, $\tilde{x}$, is to solve

$$\begin{cases} \tilde{x} = x_0 + V_m\hat{x} \\ (AV_m)^T(b - A\tilde{x}) = 0 \end{cases}.$$

Setting $W_m = AV_m$, and simplifying gives

$$\begin{cases} \tilde{x} = x_0 + V_m\hat{x} \\ \left(W_m^T W_m\right)\hat{x} = \|r_0\| W_m^T v_1 \end{cases}.$$

Assuming, no breakdown occurs we have $\text{span}\{v_0, \ldots, v_{m-1}\} = \text{span}\{v_1, Av_1, \ldots, Av_{m-1}\}$, and so $W_{m-1} = AV_{m-1}$ has linearly independent columns. If we further assume $Av_{m-1} \notin \mathcal{K}_m$, then we get W_m is invertible and the above system can be simplified to

$$\begin{cases} \tilde{x} = x_0 + V_m\hat{x} \\ \hat{x} = \|r_0\| W_m^{-1} v_1 \end{cases}. \tag{3.10}$$

3.2.8 Practical Implementation Details

Practical implementation details of the Krylov subspace methods are quite important, however, they have not been within our purview thus far. For example, Paige and Saunders' SYMMLQ, which is a Lanczos method, was not the first theoretically sound method proposed to deal with symmetric indefinite systems [93, 43, 79, 80], however, it has been more widely adopted due its practical superiority [93, 104, 105]. Two practical implementation details of GMRES that are hard to avoid even in a more theoretical discussion are restarts and preconditioners. The latter is in fact essential to all iterative methods and is discussed in Section 3.3. Before delving into their discussion, since the discussion of practical implementation details has come about,

we briefly mention a few details that were omitted earlier.

First of all, the orthogonalization process used in Arnoldi's algorithm is usually replaced with one more resembling the modified Gram-Schmidt process, which replaces the single projection onto $(\text{span}\{v_1, \ldots, v_j\})^\perp$ with j iterative projections onto $(\text{span}\{v_1\})^\perp, (\text{span}\{v_2\})^\perp, \ldots, (\text{span}\{v_j\})^\perp$, or even Householder orthogonalization for greater stability [104]. For example, most formulations of the Lanczos method discussed in Section 3.2.3 uses the modified Gram-Schmidt process, which we include below.

Algorithm 6 Lanczos Algorithm (modified Gram-Schmidt version)

Require: arbitrary unit vector v_1
  Define $\beta_1, v_0 = 0$
  for $j = 1, 2, \ldots, m$ **do**
    $\omega_j = Av_j - \beta_j v_{j-1}$ % projection onto $(\text{span}\{v_{j-1}\})^\perp$
    $\alpha_j = \langle \omega_j, v_j \rangle$
    $\omega_j = \omega_j - \alpha_j v_j$ % second projection onto $(\text{span}\{v_j\})^\perp$
    $\beta_{j+1} = \|\omega_j\|$
    if $\beta_{j+1} = 0$ **then**
      Stop
    end if
    $v_{j+1} = \frac{1}{\beta_{j+1}} \omega_j$ % normalizing
  end for

In addition, most implementations of GMRES do not obtain the approximation by directly imposing (3.10). Instead, the approximation, $\tilde{x}$, is picked so that $\|b - A\tilde{x}\|$ is minimized, giving rise to the name "minimal residual". To the uninitiated, this may seem more difficult than our previous approach, but in fact it turns out to be a $(m+1) \times m$ least squares problem, where m is relatively small. To see this, we first define $\tilde{H}_m$ to be the $(m+1) \times m$ Hessenberg matrix defined by Arnoldi's algorithm. That is, $\tilde{H}_k$ is H_k with the row $h_{(k+1)k} e_k^T$ appended at the bottom. It is not too hard to see, from (3.5), that $AV_k = V_{k+1}\tilde{H}_k$. As seen previously, if $\tilde{x} \in x_0 + \mathcal{K}_m$, we obtain

$$\|b - A\tilde{x}\| = \|r_0 - AV_m\hat{x}\|$$

$$= \left\| \|r_0\| v_1 - V_{m+1} \tilde{H}_m \hat{x} \right\|$$
$$= \left\| V_{m+1} \Big(\|r_0\| e_1 - \tilde{H}_m \hat{x} \Big) \right\|.$$

So to find $\hat{x}$, and thus $\tilde{x}$, we must minimize $\left\| V_{m+1} \Big(\|r_0\| e_1 - \tilde{H}_m x \Big) \right\|$. As $V_{m+1}^T V_{m+1} = I_{m+1}$, this is tantamount to minimizing $\left\| \|r_0\| e_1 - \tilde{H}_m x \right\|$, which is an $(m+1) \times m$ least squares problem. We now state the original and most common form of the GMRES method.

Algorithm 7 GMRES (modified Gram-Schmidt version)

Require: x_0 initial approximation for solution of (3.1)
  Define $r_0 = b - Ax_0, v_1 = \frac{1}{\|r_0\|} r_0$
  for $j = 1, 2, \ldots, m$ **do**
    $\omega_j = Av_j$
    for $i = 1, 2, \ldots, j$ **do**
      $h_{ij} = \langle \omega_j, v_i \rangle$
      $\omega_j = \omega_j - h_{ij} v_i$ % iterative projections onto $(\text{span}\{v_i\})^\perp$ for $i = 1, 2, \ldots, j$
    end for
    $h_{(j+1)j} = \|\omega_j\|$
    if $h_{(j+1)j} = 0$ **then**
      $m = j$
      Break
    end if
    $v_{j+1} = \dfrac{1}{h_{(j+1)j}} \omega_j$ % normalizing
  end for
  $\hat{x} = \arg\min_{x \in \mathbb{R}^m} \left\| \|r_0\| e_1 - \tilde{H}_m x \right\|$ % minimize $\|r\| = \|b - A\tilde{x}\|$
  $\tilde{x} = x_0 + V_m \hat{x}$ % $\tilde{x} \in x_0 + \mathcal{K}_m$

The minimization step can be solved using plane rotations which iteratively transform the Hessenberg matrix into an upper triangular matrix [105, 104]. So, like the Lanczos method, GMRES can also be posed as an iterative method.

Restarts

Due to the high cost and memory requirement of the Gram-Schmidt orthogonalization process, GMRES gets prohibitively expensive as m, the dimension of the Krylov

subspace $\mathcal{K}_m$, gets large. However, since our goal is to find a solution to (3.1), it appears that we should try to keep m as large as possible so we end up finding that solution. The key realization to solving these seemingly contradicting objectives is that the Krylov subspaces $\mathcal{K}_m(r_0)$ are nested for the same initial residual vector only. That is $\mathcal{K}_\ell(r_0) \subset \mathcal{K}_m(r_0)$ whenever $\ell \leq m$ for the same residual vector but, in general $\mathcal{K}_m(r_0) \neq \mathcal{K}_m(r_0')$. So it is true, as expected, that the m^{th} approximation, $\tilde{x}_m$, is a better (or identical) approximation to the solution of (3.1) than $\tilde{x}_\ell$, for $\ell \leq m$. However, instead of looking at Krylov subspaces of prohibitively high dimension in order to improve our approximations, we may be able to get away with looking at different Krylov subspaces of low dimensions. Since approximations only get better (or stay the same), we do not risk getting worse approximations by changing the Krylov subspace we are projecting onto. Thus, given an initial residual vector, r_0, we iteratively look for a suitable solution to (3.1) in the nested Krylov subspaces $\mathcal{K}_1(r_0) \subset \ldots \subset \mathcal{K}_m(r_0)$ for a suitable value of m. If we are unsuccessful, we repeat the same process but starting with the m^{th} residual vector obtained from projecting onto $\mathcal{K}_m(r_0)$. Going from projecting onto the m-dimensional subspace $\mathcal{K}_m(r_0)$ to the 1-dimensional subspace $\mathcal{K}_1(r_0')$ is called restarting [105]. GMRES(m) refers to the GMRES procedure with restarting every m steps [105] and it is stated below.

In general, choosing an appropriate restart value is problem dependent and is more of an art than a science [11]. On the one hand, it is easy to see that GMRES with larger restart values has a higher chance of finding an appropriate approximation to the solution, while with small restart values it may stagnate and never find such an approximation. On the other hand, it has been shown that in some cases the opposite is true; GMRES with lower restart values found an appropriate approximation of the solution in fewer iterations than higher ones, which at times stagnated and never found an appropriate approximation [39, 38].

Algorithm 8 GMRES(m) (modified Gram-Schmidt version)

Require: x_0 initial approximation for solution of (3.1)
  Define $r_0 = b - Ax_0, v_1 = \frac{1}{\|r_0\|} r_0$
  while not converge **do**
    for $j = 1, 2, \ldots, m$ **do**
      $\omega_j = Av_j$
      for $i = 1, 2, \ldots, j$ **do**
        $h_{ij} = \langle \omega_j, v_i \rangle$
        $\omega_j = \omega_j - h_{ij} v_i$ % iterative projections onto $(\text{span}\{v_i\})^{\perp}$ for $i = 1, 2, \ldots, j$
      end for
      $h_{(j+1)j} = \|\omega_j\|$
      if $h_{(j+1)j} = 0$ **then**
        $m = j$
        Break
      end if
      $\hat{x}_j = \arg\min_{x \in \mathbb{R}^m} \left\| \|r_0\| e_1 - \tilde{H}_j x \right\|$ % minimize $\|r_j\| = \|b - A\tilde{x}_j\|$
      $\tilde{x}_j = x_0 + V_j \hat{x}_j$ % $\tilde{x}_j \in x_0 + \mathcal{K}_j$
      if $\tilde{x}_j$ is a suitable approximation **then**
        $\tilde{x} = \tilde{x}_j$ % the desired approximation is found
        Break
      end if
      $v_{j+1} = \dfrac{1}{h_{(j+1)j}} \omega_j$ % normalizing
    end for
    $r_0 = r_m$
    $v_1 = \frac{1}{\|r_0\|} r_0$
  end while

3.3 Preconditioners

Unlike analytical solutions, finding numerical solutions to a linear system, say (3.1), depends on various numerical properties of the matrix A, such as its spectral properties, condition number, singular values, sparsity pattern, and bandwidth [104, 119, 11, 22]. Preconditioning a linear system is replacing it with one that has the same solution set but with better numerical properties. From that perspective, preconditioning can be used for both direct and iterative solvers. However, since preconditioning is essential to iterative methods, most discussions of preconditioning are usually restricted to iterative solvers [104, 119, 11]. In fact, many attribute the rise of iterative solvers in the 1970s and 1980s to the developments in preconditioning techniques [104, 119].

Historically, the concept of preconditioning linear systems is well over 150 years old. Indeed, in 1845 Jacobi [61] described preconditioning linear systems, which arise from least square problems, to make them more diagonally dominant [13, 17]. However, the word "preconditioning" seems to originate with Turing [122] in 1948 [119, 13]. The link between preconditioning and iterative methods, specifically Krylov subspace methods, came much later around 1970s. The highly influential paper by Meijerink and van der Vorst [85], in which they introduced an incomplete factorization preconditioner to the method of conjugate gradients, is credited with bringing much of the attention to this connection between preconditioners and iterative methods [104, 119, 13]. It is not that they were the first to introduce incomplete factorization, rather it is that they demonstrated the potency of this combination [104, 13].

In general, preconditioning a linear system, say (3.1), is done via a nonsingular matrix M, referred to as the preconditioner. The preconditioner can be applied as a left preconditioner,

$$M^{-1}Ax = M^{-1}b,$$

a right preconditioner,

$$AM^{-1}y = b,\ x = M^{-1}y,$$

or, if we write $M = M_1M_2$, a split preconditioner,

$$M_1^{-1}AM_2^{-1}y = M_1^{-1}b,\ x = M_2^{-1}y.$$

The ultimate goal here is that the new system is better to solve than the original one (better here can be thought of as cheaper to solve or yields a more accurate answer when solved). From that perspective, choosing $M = A$ is probably the best idea. Further investigating this idea, gets us into the practical restraint that we must satisfy, namely, the preconditioner must be relatively easy to apply. In the $M = A$ example, applying the preconditioner is as difficult as solving the original system, which is of no benefit. On the other hand, choosing $M = I$ is the easiest preconditioner to apply, however, the new system is not better to solve than the original one. A balance between the goal and restraint of preconditioning, which is usually problem dependent, is where most of the useful preconditioners are found. Typically, M is chosen so that it approximates A in some sense so that the matrix $M^{-1}A$, for example, is easier to solve with than A.

In some cases, M^{-1}, instead of M, is known and thus applying the preconditioner is as simple as matrix multiplication. Some use this to define a new type of preconditioners, the inverse type preconditioners [22]. An inverse type preconditioner is a matrix, M, which is usually chosen to approximate A^{-1} instead of A. Inverse type preconditioners can also be applied as left, right, or split preconditioners in much the same way shown before, for example

$$MAx = Mb$$

is a left inverse type preconditioners. This distinction of the types of preconditioners is of little practical consequences since, in practice, preconditioners can either be applied through solving a linear system or matrix multiplication, as mentioned earlier. This along with the fact that most preconditioners happen to be of the first type introduced [11], make it clear why most discussions of preconditioners omit this distinction. In the remainder of this section, we will survey preconditioners from the scope of iterative solvers. We will mainly focus the ones we found success with in the GMRES iteration for solving the linear systems arising in our simulation (see Section 4.3.2). For more detailed discussions, the reader is referred to [104, 119, 11, 22].

3.3.1 Incomplete LU Factorization

An LU factorization of a matrix $A = LU$, is obtained via Gaussian elimination so that L and U are, respectively, lower and upper triangular nonsingular matrices. In the case of large and sparse matrices, Gaussian elimination may result in significantly more dense factors than the original matrix. For example, Fig. 3.1 shows an example in which the L, U factors are at least twice as dense as the original matrix.

An ILU factorization of a matrix A is a lower triangular matrix L and an upper triangular matrix U, such that $A = LU+R$, for some remainder matrix R. In practice, ILU factorizations are rarely computed as an LU factorization of $A - R$. Rather, it is obtained by dropping, meaning setting to zero, some nonzero entries during the Gaussian elimination process as it is applied to A (refer to, for example, [104, 22] for more details). The different variations of ILU preconditioners mostly arise from which elements are dropped and how the dropping decisions are made [104, 22].

The simplest ILU preconditioner is ILU(0), in which an element of L and U is dropped if it corresponds to a zero entry in the original matrix A. Fig. 3.1 shows an example of this factorization including the resulting remainder matrix, R. In comparison to the LU factorization of the same matrix, the L and U factors from

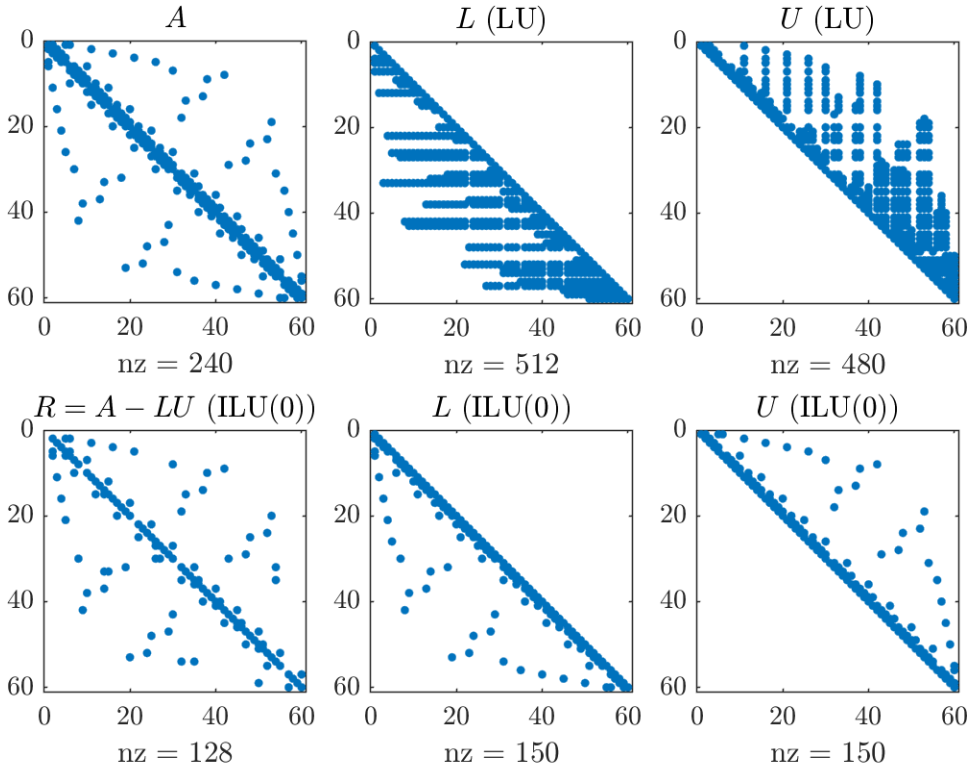


Figure 3.1: Top: Sparsity pattern of a matrix A, (left), and its L, (middle), and U, (right), factors. The LU factors are at least twice as dense as the original matrix. Bottom: The sparsity pattern of the resulting remainder matrix $R = A - LU$, (left), and L, (middle), and U, (right), factors of the ILU(0) incomplete LU factorization of A. The incomplete LU factors are more sparse than the original matrix and significantly more sparse than the complete LU factors. nz denotes the number of nonzero entries in a matrix.

ILU(0) are much more sparse. Another advantage to the ILU(0) preconditioner is that it is inexpensive to compute [104, 13]. However, for some problems the ILU(0) preconditioner is not an adequate approximation of A and the preconditioned iteration fails to converge in a reasonable number of iterations [104].

One way to obtain a better approximation of A, is to be less stringent and allow more fill-in than the sparsity pattern of A. This generalization leads to the ILU(p) preconditioners, where p is the level-of-fill allowed ($p = 0$ matches the already introduced ILU(0)). Another generalization is to drop elements in L and U whose magnitude is below a certain (relative) threshold, otherwise known as the ILUT preconditioner. A remarkable advantage to this ILU preconditioner is that it is sensitive to the numerical values rather than just the structure of A. However, it is much more expensive to apply as we discuss in Section 4.3.2. The reader is referred to [104] for a more detailed discussion of the various ILU preconditioners.

3.3.2 Reordering and Scaling

As we mentioned, the sparsity pattern and bandwidth of a coefficient matrix affect one's ability to find numerical solutions to its linear system. Thus, one should expect that reordering, or permuting, the rows and columns of a matrix can be a useful preconditioning technique. More precisely, reordering a linear system, say (3.1), is accomplished via two permutation matrices P and Q

$$PAQy = Pb,\ x = Qy. \tag{3.11}$$

A permutation matrix here denotes a product of row-interchanging elementary matrices. The permutation matrix P permutes the rows of A, while Q permutes the columns of A. Using our previous terminology, one can say that reordering is applied as a split preconditioner, with the left preconditioner permuting the equations and

the right preconditioner permuting the unknowns of the linear system.

Our discussion will be limited to a special type of premutation known as a symmetric permutation. These permutations are mostly used to reduce the fill-in that occurs when Gaussian elimination is applied on a matrix or to reduce the bandwidth of a matrix [104, 13]. These permutations, otherwise known as symmetric reorderings, are reorderings where $Q = P^T$ [104, 13]. Essentially the unknowns of the linear system are reordered in the same manner as its equations. A special property of symmetric permutation is that it leaves the adjacency graph of a matrix unchanged, except for relabelling of the vertices [104]. Thus, if a matrix has some desirable properties, such as diagonal dominance or symmetry, symmetric permutations preserve these properties while minimizing fill-in or reducing the bandwidth.

One of the more commonly used symmetric reordering techniques is the reverse Cuthill McKee (RCM) ordering [50, 104, 13]. Discovered and named by George in his thesis [49], the RCM ordering is in fact the Cuthill-McKee (CM) ordering [29] in reverse order. George observed that the RCM ordering is superior to the CM ordering in systems arising from finite element methods [49]. This observation was corroborated and generalized by other numerical experiments [28, 78] and proofs [78]. Various similar justifications to this observation have been proposed [49, 28, 78, 104], the main point of which is illustrated in Fig. 3.2. Figure 3.2 shows a comparison between the CM and RCM orderings of a matrix. While the bandwidth of the resulting matrices are identical, the CM ordering, by construction (see [29, 104, 49] for details), results in nonzero entries to the right and below the diagonal entries. This results in a significant amount of fill-in during Gaussian elimination. Reversing this ordering leads to the nonzero entries being to the left and above the diagonal entries, which results in significantly less fill-in. This was effectively distilled by Saad [104] by using the term arrow submatrices. In Fig. 3.2, we observe many arrow submatrices that point upwards in the CM ordering as compared to downwards in the RCM ordering.

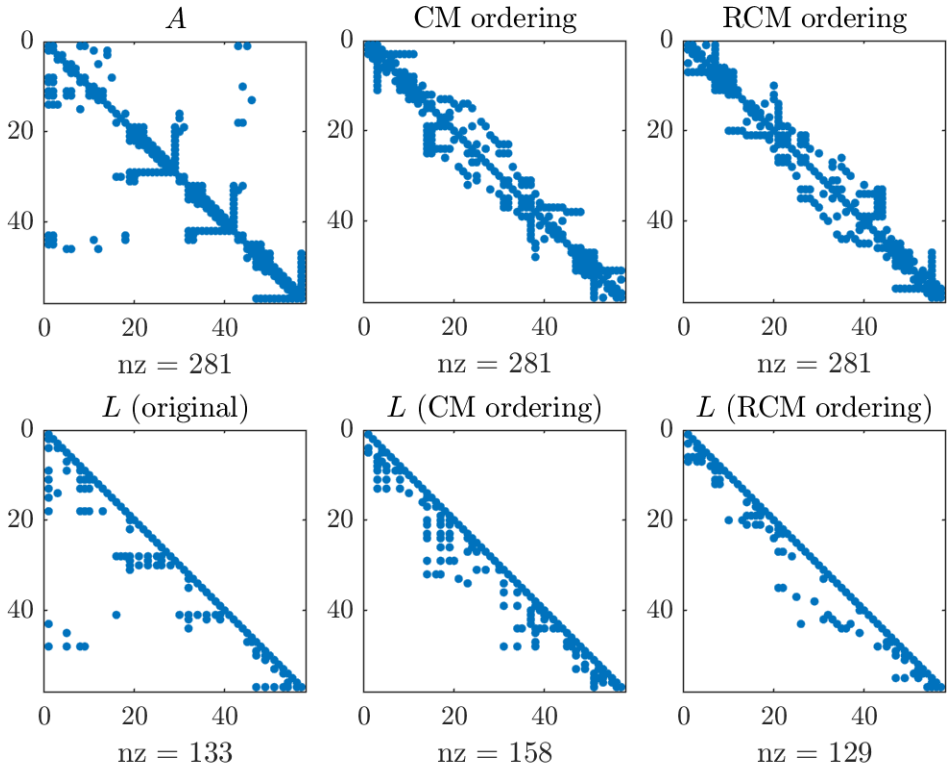


Figure 3.2: Top: Sparsity pattern of a matrix A, (left), and its CM, (middle), and RCM, (right), orderings. In the CM ordering there are many instances where nonzero entries occur to the right and below diagonal entries. In contrast, these instances correspond to nonzero entries to the left and above diagonal entries in the RCM ordering. This significantly reduces fill-in. Bottom: Sparsity pattern of the L factor (from LU factorization) of A, (left), and its CM, (middle), and RCM, (right), orderings. While the CM ordering decreased the bandwidth of A, it increased the fill-in resulting from Gaussian elimination. In contrast, RCM ordering decreased both the bandwidth and fill-in of A.

The consequence of this is significantly less fill-in for the RCM ordering. It is worth noting that the CM ordering was designed with the goal of reducing the bandwidth of a matrix [29, 104, 49, 50]. However, the RCM ordering is superior to the CM ordering as it better minimizes fill-in while still achieving the same level in reducing bandwidth.

Much like reordering, scaling of a linear system, say (3.1), can also be applied as a split preconditioner

$$RACy = Rb,\ x = Cy, \tag{3.12}$$

in which R and C are diagonal matrices scaling the rows and columns of A, respectively. Scaling can be an effective preconditioning technique considering how the numerical stability of methods such as Gaussian elimination are dependent on the size, and relative size, of the values of the matrix [119]. We provide an example of the significant impact a scaling preconditioner can have using our model in Section 4.3.2.

The combination of a reordering and scaling preconditioner has been shown to be quite effective [92, 37, 36]. A reordering and scaling preconditioner can be applied by combining (3.11) and (3.12)

$$RPAQCy = RPb,\ x = QCy,$$

in which the matrices are as defined in (3.11) and (3.12). Clearly this is equivalent to applying the scaling preconditioner first as $P^T RP$ is a diagonal matrix.

Chapter 4

Numerical Methods for Retinal Dynamics

In this chapter, we discuss the numerical methods we employed to solve the PDE system (2.3) to (2.6). For the purpose of clarity and to avoid any repetition, the discussion here will be focused on the bi-domain equations. The method can naturally be extended to the multi-domain equations.

4.1 Spatial Discretization

We discretize our system in space using non-uniform finite differences. We use a spherical coordinate system (r, θ, φ), where r is the distance of a point to the origin, θ is the polar angle in the xy-plane, and φ is the signed latitude from the xy-plane.

4.1.1 Non-Uniform Tensor Product Grid

For our finite difference discretization we opt to use a non-uniform tensor product grid (see Fig. 4.1). We choose a non-uniform grid as the retina, being an active domain which receives a variety of light stimuli, requires a very fine grid. Extending such a

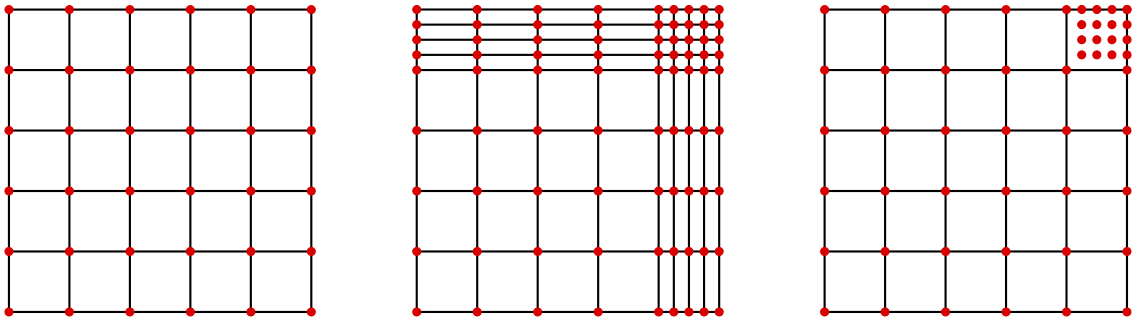


Figure 4.1: An illustrative example of the difference between uniform, non-uniform tensor product, and completely non-uniform grids. Left: Uniform grid discretization of $[0,1]^2$. The spacing is uniform of size 0.2 units. Middle: Non-uniform tensor product grid discretization of $[0,1]^2$. The grid can be obtained by the tensor product $\{0, 0.2, \ldots, 0.8, 0.85, 0.9, \ldots, 1\} \times \{0, 0.2, \ldots, 0.8, 0.85, 0.9, \ldots, 1\}$. Right: Completely non-uniform grid discretization of $[0,1]^2$. The points at the top right corner do not align with any of the gridlines making it impossible to obtain this grid as a tensor product.

grid to the rest of the eye, noting the difference in size, would make the computations unwieldy. A completely non-uniform grid (i.e. one that is not a tensor product) would be even more efficient, but difficult to implement correctly, especially considering the variety of boundary conditions in the system.

4.1.2 Discretized PDE System in Spherical Coordinates

The gradient in spherical coordinates is

$$\nabla = \bar{r}\frac{\partial}{\partial r} + \frac{1}{r\cos\varphi}\bar{\theta}\frac{\partial}{\partial \theta} + \frac{1}{r}\bar{\varphi}\frac{\partial}{\partial \varphi}, \tag{4.1}$$

in which $\bar{r}, \bar{\theta}, \bar{\varphi}$ are unit vectors in the radial, polar, and latitudinal directions, respectively, which locally form a spherical basis on $\mathbb{R}^3$ (not including the z-axis). Considering the column packing of the retina, we restricted the types of conductivity

tensors, in this spherical basis, to be of the form

$$\mu_{\mathrm{x}} = \begin{bmatrix} \mu_{\mathrm{x}}^{r} & 0 & 0 \\ 0 & \mu_{\mathrm{x}}^{\theta} & 0 \\ 0 & 0 & \mu_{\mathrm{x}}^{\varphi} \end{bmatrix} \qquad \text{for x} = \text{ i, e,}$$

in which $\mu_{\mathrm{x}}^{r}, \mu_{\mathrm{x}}^{\theta}, \mu_{\mathrm{x}}^{\varphi}$ are the conductivities in the radial, polar, and latitudinal directions, respectively. Off of the retina, on $\mathcal{S} \setminus \mathcal{R}$ (Section 2.1), considering we are in the assumed electrically passive vitreous chamber, in which current has equal resistance for traveling in all directions (isotropic), one can justifiably consider μ_s to be a non-negative scalar (equivalently, $\mu_{\mathrm{s}}^{r} = \mu_{\mathrm{s}}^{\theta} = \mu_{\mathrm{s}}^{\varphi}$). Continuing generically, locally we can write μ_{x} as a linear transformation

$$\mu_{\mathrm{x}}(r,\theta,\varphi)(v) = \mu_{\mathrm{x}}^{r}(r,\theta,\varphi)\bar{r}^{*}(v)\bar{r} + \mu_{\mathrm{x}}^{\theta}(r,\theta,\varphi)\bar{\theta}^{*}(v)\bar{\theta} + \mu_{\mathrm{x}}^{\varphi}(r,\theta,\varphi)\bar{\varphi}^{*}(v)\bar{\varphi} \qquad \text{for x} = \text{ i, e, s,} \quad (4.2)$$

in which $\bar{r}^{*}, \bar{\theta}^{*}, \bar{\varphi}^{*}$ are the standard linear functionals associated with the basis $\bar{r}, \bar{\theta}, \bar{\varphi}$. Using (4.1) and (4.2), we can express (2.3) to (2.5) in spherical coordinates. For example, (2.3) can be written in spherical coordinates as

$$\frac{\partial \mu_{\mathrm{s}}^{r}}{\partial r}\frac{\partial \phi_{\mathrm{s}}}{\partial r} + \mu_{\mathrm{s}}^{r}\frac{\partial^2 \phi_{\mathrm{s}}}{\partial r^2} + 2\frac{\mu_{\mathrm{s}}^{r}}{r}\frac{\partial \phi_{\mathrm{s}}}{\partial r} + \frac{1}{r^2\cos^2\varphi}\left(\frac{\partial \mu_{\mathrm{s}}^{\theta}}{\partial \theta}\frac{\partial \phi_{\mathrm{s}}}{\partial \theta} + \mu_{\mathrm{s}}^{\theta}\frac{\partial^2 \phi_{\mathrm{s}}}{\partial \theta^2}\right) - \frac{\mu_{\mathrm{s}}^{\varphi}\sin\varphi}{r^2\cos\varphi}\frac{\partial \phi_{\mathrm{s}}}{\partial \varphi} + \frac{1}{r^2}\left(\frac{\partial \mu_{\mathrm{s}}^{\varphi}}{\partial \varphi}\frac{\partial \phi_{\mathrm{s}}}{\partial \varphi} + \mu_{\mathrm{s}}^{\varphi}\frac{\partial^2 \phi_{\mathrm{s}}}{\partial \varphi^2}\right) = 0. \quad (4.3)$$

The spherical coordinate representation of the other equations is included in Appendix B.

We use the non-uniform centered finite difference formulas to discretize the first and second derivatives. Let $\{r^0 = 0, \ldots, r^n = r_{\mathrm{eye}}\}$ be the discretization points in the radial direction and $\phi^{(i,j,k)}$ correspond to the value of ϕ at the (i,j,k) node, where

i, j, k are indices for the radial, polar, and latitudinal directions, respectively. The equations we use are

$$\frac{\partial \phi^{(i,j,k)}}{\partial r} = \frac{\phi^{(i+1,j,k)} - \gamma_i^2 \phi^{(i-1,j,k)} - (1-\gamma_i^2)\phi^{(i,j,k)}}{(1+\gamma_i)h_{i+1}}, \tag{4.4}$$

$$\frac{\partial^2 \phi^{(i,j,k)}}{\partial r^2} = 2\gamma_i \frac{\phi^{(i+1,j,k)} + \gamma_i \phi^{(i-1,j,k)} - (1+\gamma_i)\phi^{(i,j,k)}}{(1+\gamma_i)h_{i+1}^2}, \tag{4.5}$$

in which $h_{i+1} = r^{i+1} - r^i$ and $\gamma_i = \frac{h_{i+1}}{h_i}$. The finite difference equations used for the other variables are also included in Appendix B.

Equations (4.3) to (4.5) (and similar equations for the polar and latitudinal directions) provide a linear relation between the potential values on the grid. From (2.3) and boundary conditions (2.1) and (2.2), we have

$$A\boldsymbol{\phi}_\mathbf{s} = B\boldsymbol{\phi}_\mathbf{e}, \tag{4.6}$$

in which A and B are sparse matrices, and $\boldsymbol{\phi}_\mathbf{s}$ and $\boldsymbol{\phi}_\mathbf{e}$ are vectors containing the grid point values of the corresponding functions stored in reverse lexicographical order. Similarly, (2.4) becomes

$$C[\boldsymbol{\phi}_\mathbf{s}; \boldsymbol{\phi}_\mathbf{e}; \boldsymbol{V}_\mathbf{m}] = \mathbf{0}, \tag{4.7}$$

in which C is also a sparse matrix, $\boldsymbol{V}_\mathbf{m}$ is a vector containing the grid point values of V_m stored in reverse lexicographical order, and the square bracket notation $[\,\cdot\,;\,\cdot\,]$ denotes concatenating two or more vectors. Unambiguously, given two vectors $\boldsymbol{x} = (x_1, \ldots, x_n), \boldsymbol{y} = (y_1, \ldots, y_m)$, using the square bracket notation we get $[\boldsymbol{x}; \boldsymbol{y}] = (x_1, \ldots, x_n, y_1, \ldots, y_m)$. Equations (2.5) and (2.6) become a system of ODEs for the values of V_m and $\boldsymbol{X}$ on the grid,

$$\frac{d[\boldsymbol{V}_\mathbf{m}; \boldsymbol{X}]}{dt} = \boldsymbol{G}(\boldsymbol{\phi}_\mathbf{s}, \boldsymbol{\phi}_\mathbf{e}, \boldsymbol{V}_\mathbf{m}, \boldsymbol{X}). \tag{4.8}$$

Note that we abuse the notation by writing $\boldsymbol{X}$ for both the function and its values on the grid. We also note that, in this setup, $\boldsymbol{G}$ is a nonlinear function as a result of the nonlinearity in the transmembrane currents, I_{m}. Equations (4.6) to (4.8) are a system of differential algebraic equations (DAEs). A summary of the theory of DAEs and methods for their numerical solution is described in [9].

4.1.3 Boundary Conditions

To setup the detailed discussion of the boundary conditions , let $\{r^0 > 0, \ldots, r^i = r_{\mathrm{eye}} - r_{\mathrm{retina}}, \ldots, r^n = r_{\mathrm{eye}}\}$ be the grid lines in the radial direction. We do not include $r = 0$ in our grid as it is a coordinate singularity. When the finite differences require the value at the origin, it is given by taking the average values of grid points on the sphere of radius r^0.

The potential ϕ_{e} appears in (4.6) as a result of the retinal boundary conditions imposed on the system. In fact, the discretization resulting in (4.6) to (4.8) incorporates the boundary conditions. The Neumann boundary conditions in our system are

$$n_{\mathrm{x}} \cdot (\mu_{\mathrm{i}} \nabla V_{\mathrm{m}}) = -n_{\mathrm{x}} \cdot (\mu_{\mathrm{i}} \nabla \phi_{\mathrm{e}}) \qquad (\text{on } \partial\mathcal{R}_{\mathrm{x}} \text{ for } \mathrm{x} = \mathrm{o}, \mathrm{i}, \ell), \tag{4.9}$$

$$n_{\mathrm{x}} \cdot (\mu_{\mathrm{e}} \nabla \phi_{\mathrm{e}}) = n_{\mathrm{x}} \cdot (\mu_{\mathrm{s}} \nabla \phi_{\mathrm{s}}) \qquad (\text{on } \partial\mathcal{R}_{\mathrm{x}} \text{ for } \mathrm{x} = \mathrm{i}, \ell), \tag{4.10}$$

$$n_{\mathrm{o}} \cdot (\mu_{\mathrm{e}} \nabla \phi_{\mathrm{e}}) = 0 \qquad (\text{on } \partial\mathcal{R}_{\mathrm{o}}), \tag{4.11}$$

$$n_{\mathrm{s}} \cdot (\mu_{\mathrm{s}} \nabla \phi_{\mathrm{s}}) = 0 \qquad (\text{on } \partial\mathcal{S} \setminus \partial\mathcal{R}_{\mathrm{o}}), \tag{4.12}$$

which were obtained from the original boundary conditions by using $\phi_{\mathrm{i}} = V_{\mathrm{m}} + \phi_{\mathrm{e}}$, to eliminate ϕ_{i}.

The conditions in (4.11) and (4.12) are tantamount to $\frac{\partial \phi}{\partial r} = 0$ (we dropped the subscript as the treatment of (4.11) and (4.12) are similar). We impose this condition

using the grid extension method [76] to get

$$\frac{\partial \phi^{(n,j,k)}}{\partial r} = \frac{\phi^{(n+1,j,k)} - \gamma_n^2 \phi^{(n-1,j,k)} - (1-\gamma_n^2)\phi^{(n,j,k)}}{(1+\gamma_n)h_{n+1}} = 0,$$

in which $\gamma_n = 1$ as we choose $h_{n+1} = h_n$. Thus we get $\phi^{(n+1,j,k)} = \phi^{(n-1,j,k)}$, which gives that the second order radial derivative is

$$\frac{\partial^2 \phi^{(n,j,k)}}{\partial r^2} = 2\gamma_n \frac{\phi^{(n+1,j,k)} + \gamma_n \phi^{(n-1,j,k)} - (1+\gamma_n)\phi^{(n,j,k)}}{(1+\gamma_n)h_{n+1}^2} = \frac{2\phi^{(n-1,j,k)} - 2\phi^{(n,j,k)}}{h_n^2}.$$

Now we deal with the first group of boundary conditions, and we take (4.10), with x = i, as our working example. After expansion, (4.10) becomes $\mu_{\mathrm{e}}^r \frac{\partial \phi_{\mathrm{e}}}{\partial r} = \mu_{\mathrm{s}}^r \frac{\partial \phi_{\mathrm{s}}}{\partial r}$ (on ∂R_i). Using the one sided finite difference formula on ϕ_{s} and ϕ_{e} we get

$$\mu_{\mathrm{e}}^r \frac{\phi_{\mathrm{e}}^{(i,j,k)} - \phi_{\mathrm{e}}^{(i-1,j,k)}}{h_i} = \mu_{\mathrm{s}}^r \frac{\phi_{\mathrm{s}}^{(i,j,k)} - \phi_{\mathrm{s}}^{(i-1,j,k)}}{h_i},$$

which can be rearranged to give

$$\phi_{\mathrm{e}}^{(i-1,j,k)} = \frac{\mu_{\mathrm{s}}^r}{\mu_{\mathrm{e}}^r}\phi_{\mathrm{s}}^{(i-1,j,k)} + \left(1 - \frac{\mu_{\mathrm{s}}^r}{\mu_{\mathrm{e}}^r}\right)\phi_{\mathrm{e}}^{(i,j,k)}, \tag{4.13}$$

as $\phi_{\mathrm{e}}^{(i,j,k)} = \phi_{\mathrm{s}}^{(i,j,k)}$ on $\mathcal{R}_{\mathrm{i}}$. Again using the grid extension method, (4.13) can be used in (4.4) and (4.5) to compute the derivative on $\mathcal{R}_{\mathrm{i}}$, which finishes our treatment of the boundary conditions.

4.2 An Adaptive Time-stepper

After spatial discretization, we must solve the DAE in time. This involves solving an equation of the form $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{s}, t, \boldsymbol{x})$. To simplify notation, we can assume that we are interested in $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(t, \boldsymbol{x})$ instead, since we can solve for the potentials from the membrane voltage using (4.6) and (4.7). However, we emphasize that in our

implementation we solve the system in its original form, (4.6) to (4.8), to preserve its sparsity [46]. To solve the differential equation, we use the second order, variable step-size, backward differentiation formula (BDF2). That is, to solve for the value of $\boldsymbol{x}$ at time $t_{n+1} = t_n + h_n$ (i.e. $\boldsymbol{x}_{n+1}$) we must solve the equation

$$\boldsymbol{x}_{n+1} + \alpha_0 \boldsymbol{x}_n + \alpha_1 \boldsymbol{x}_{n-1} = h_n \beta \boldsymbol{f}(t_{n+1}, \boldsymbol{x}_{n+1}),$$

with parameters $\alpha_0, \alpha_1, \beta$ given later in (4.14). We do so using Newton's method to find the root of the function $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{x} + \alpha_0 \boldsymbol{x}_n + \alpha_1 \boldsymbol{x}_{n-1} - h_n \beta \boldsymbol{f}(t_{n+1}, \boldsymbol{x})$. So the linear equation we need to solve is $\boldsymbol{F}'(\boldsymbol{x}^i)\Delta\boldsymbol{x}^{i+1} = -\boldsymbol{F}(\boldsymbol{x}^i)$, in which $\boldsymbol{x}^i$ is the value of the previous Newton iteration and $\Delta\boldsymbol{x}^{i+1} = \boldsymbol{x}^{i+1} - \boldsymbol{x}^i$. This equation and the algebraic constraints (4.6) and (4.7) can be written as

$$D^i \Delta \boldsymbol{x}^{i+1} = \boldsymbol{r}^i,$$

in which

$$\boldsymbol{r}^i = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ -\boldsymbol{F}(\boldsymbol{x}^i) \\ 0 \end{bmatrix}$$

and D^i is an $(n+1) \times n$ matrix composed of an arrangement of A, B, C, and the Jacobian $\boldsymbol{F}'(\boldsymbol{x}^i)$. The extra row at the bottom of D^i and $\boldsymbol{r}^i$ is to pick the additive constant for the potentials by imposing (2.7). Solving this Jacobian update equation is the most time consuming step in the simulation and we will discuss various aspects of it, including the size and structure of the Jacobian and the ordering of the variables, in great detail in Section 4.3.

It is worth noting that when light hits the retina and activates the opsin proteins,

it triggers a cascade of events bringing about rapid changes in the retina. Once the light stimulus is gone, retinal cells return to a resting state relatively slowly. For example, in the case of a 20 ms light flash, the rapid change occurs within 0.1 s of the onset of the flash, while it takes the photoreceptors around 1 s to return to their resting state. The presence of these multiple time scales in our model and the costly computation of each time-step, necessitates a variable time-step solver. The complexity of the system we are studying, as well as all the possible variability in light stimuli make it clear that we must employ an adaptive time-stepping method as opposed to a variable time-stepping method with preset time-step values.

4.2.1 General Scheme and Underlying Numerical Method

We are using this time-stepper to solve the differential equation in our DAE system, namely

$$\frac{d[\boldsymbol{V}_{\mathbf{m}}; \boldsymbol{X}]}{dt} = \boldsymbol{G}(\boldsymbol{\phi}_{\mathbf{s}}, \boldsymbol{\phi}_{\mathbf{e}}, \boldsymbol{V}_{\mathbf{m}}, \boldsymbol{X}) \qquad \text{(on } \mathcal{R}\text{)}.$$

To simplify notation, the details of the time-stepper will be explained as to solve the generic equation $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(t, \boldsymbol{x})$. Let $\boldsymbol{x}_n$ denote the computed approximation of $\boldsymbol{x}(t_n)$ and $h_n = t_{n+1} - t_n$ be the step-size of the n^{th} step.

The basic idea of the method is a coarse-fine computation, which has been well studied and is regularly used to study various physical systems [53, 130]. Starting at $\boldsymbol{x}_n$ we compute our coarse approximation, $\boldsymbol{x}^{\text{c}}_{n+1}$, using one step of size h_n. We then go back to $\boldsymbol{x}_n$ and compute the fine approximation, $\boldsymbol{x}^{\text{f}}_{n+1}$, using two steps of size $h_n/2$. Finally, we use both these approximations to estimate our coarse local truncation error, ϵ_{c}, (see Section 4.2.2) and if the error is suitable the step-size, h_n, and the coarse approximation are accepted. If the error is too small or too large then

we reject the time-step and choose a new step-size according the formula

$$h_n \cdot \min\left\{\max\left\{\left(\frac{tol}{\epsilon_{\mathrm{c}}}\right)^{\frac{1}{p}}, \eta_{\min}\right\}, \eta_{\max}\right\},$$

in which tol is the desired local truncation error (LTE), p is the order of the LTE (calculated in Section 4.2.2), and $\eta_{\max}, \eta_{\min}$ are safety factors that prevent h_n from drastically changing from one iteration to the next. Once a new h_n is chosen, we recompute $\boldsymbol{x}^{\mathrm{c}}_{n+1}$ and $\boldsymbol{x}^{\mathrm{f}}_{n+1}$ and repeat the procedure to get an LTE within the accepted range.

The underlying numerical method used here is the variable step-size BDF2 given by

$$\boldsymbol{x}_{n+1} - \frac{(1+\omega_n)^2}{1+2\omega_n}\boldsymbol{x}_n + \frac{\omega_n^2}{1+2\omega_n}\boldsymbol{x}_{n-1} = h_n \frac{1+\omega_n}{1+2\omega_n}\boldsymbol{f}(t_{n+1}, \boldsymbol{x}_{n+1}), \tag{4.14}$$

in which $\omega_n = \frac{h_n}{h_{n-1}}$. To compute $\boldsymbol{x}^{\mathrm{c}}_{n+1}$ we use $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n-1}$. We also use $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n-1}$ to approximate $\boldsymbol{x}$ at the half-step ($t_{n+\frac{1}{2}} = t_n + \frac{h_n}{2}$), $\boldsymbol{x}_{n+\frac{1}{2}}$. We subsequently use $\boldsymbol{x}_{n+\frac{1}{2}}$ and $\boldsymbol{x}_n$ to compute $\boldsymbol{x}^{\mathrm{f}}_{n+1}$. A schematic sketch of the method is provided in Fig. 4.2. It should be clear from Fig. 4.2 that the coarse and fine computation are completely independent and could be carried out in parallel to enhance performance. However, since there are only two parallel tasks of modest duration, any performance gains will be diminished by the overhead costs of parallelization.

4.2.2 Local Truncation Error Estimate

Let us approximate $\mathbf{LTE} = \boldsymbol{x}(t_{n+1}) - \boldsymbol{x}_{n+1}$ for our method. Using (4.14), $\boldsymbol{f}(t_{n+1}, \boldsymbol{x}_{n+1}) \approx \boldsymbol{x}'(t_{n+1})$ (see Section 5.2.2), and a Taylor series about t_n (assuming $\boldsymbol{x}_n = \boldsymbol{x}(t_n)$ and $\boldsymbol{x}_{n-1} = \boldsymbol{x}(t_{n-1})$) we get

$$\mathbf{LTE} = -\frac{1+\omega_n}{1+2\omega_n}\frac{\boldsymbol{x}'''(t_n)}{3!}h_n^2(h_n + h_{n-1}) + O(h^4), \tag{4.15}$$

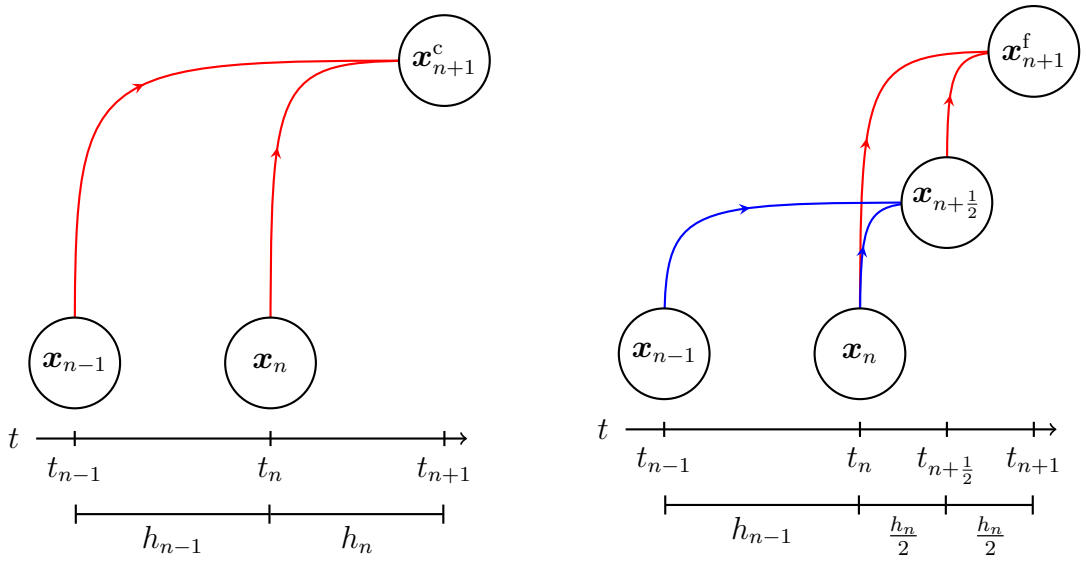


Figure 4.2: Left: The coarse approximation of $\boldsymbol{x}$ at t_{n+1}, $\boldsymbol{x}^{\mathrm{c}}_{n+1}$, is computed using two accepted values, $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n-1}$, which were precomputed at previous time-steps. Right: The fine approximation of $\boldsymbol{x}$ at t_{n+1}, $\boldsymbol{x}^{\mathrm{f}}_{n+1}$, is computed in two steps. First we compute the intermediate value $\boldsymbol{x}_{n+\frac{1}{2}}$, which approximates $\boldsymbol{x}$ at $t_{n+\frac{1}{2}} = t_n + \frac{h_n}{2}$ (shown in blue). Then we compute $\boldsymbol{x}^{\mathrm{f}}_{n+1}$ using $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n+\frac{1}{2}}$ (shown in red).

in which h^4 is understood as a product of h_n and h_{n-1} with combined powers of 4. From (4.15), we obtain the coarse LTE,

$$\boldsymbol{\epsilon}_{\mathrm{c}} = -\frac{1+\omega_n}{1+2\omega_n}\frac{\boldsymbol{x}'''(t_n)}{3!}h_n^2(h_n+h_{n-1}) + O(h^4). \tag{4.16}$$

We now compute the fine LTE, $\boldsymbol{\epsilon}_{\mathrm{f}} = \boldsymbol{x}(t_{n+1}) - \boldsymbol{x}^{\mathrm{f}}_{n+1}$. Using (4.14) with equal step-sizes $\frac{h_n}{2}$ on the second fine step we obtain

$$\boldsymbol{\epsilon}_{\mathrm{f}} = \boldsymbol{x}(t_{n+1}) - \left(\frac{4}{3}\boldsymbol{x}_{n+\frac{1}{2}} - \frac{1}{3}\boldsymbol{x}_n + \frac{h_n}{2}\frac{2}{3}\boldsymbol{f}(t_{n+1}, \boldsymbol{x}_{n+1})\right).$$

We then use (4.14) a second time but with step-size h_{n-1} and $\frac{h_n}{2}$ on $\boldsymbol{x}_{n+\frac{1}{2}}$ and obtain the fine LTE to be

$$\boldsymbol{\epsilon}_{\mathrm{f}} = -\frac{1}{3!\cdot 3}\,\frac{1}{1+\omega_n}\boldsymbol{x}'''(t_n)h_n^2h_{n-1} - \frac{1}{3!\cdot 2}\,\frac{1+\frac{\omega_n}{2}}{1+\omega_n}\boldsymbol{x}'''(t_n)h_n^3 + O(h^4).$$

The third derivative in (4.15) can be approximated using our coarse-fine approach since

$$\boldsymbol{x}^{\mathrm{c}}_{n+1} - \boldsymbol{x}^{\mathrm{f}}_{n+1} = \boldsymbol{\epsilon}_{\mathrm{f}} - \boldsymbol{\epsilon}_{\mathrm{c}} \approx \left(\frac{\frac{1}{2}\omega_n^2 + \frac{3}{4}\omega_n + \frac{1}{2}}{(1+2\omega_n)(1+\omega_n)} \frac{\boldsymbol{x}'''(t_n)}{3!} \right) h_n^3 + \left(\frac{\omega_n^2 + \frac{4}{3}\omega_n + \frac{2}{3}}{(1+2\omega_n)(1+\omega_n)} \frac{\boldsymbol{x}'''(t_n)}{3!} \right) h_n^2 h_{n-1}.$$

Solving for $\boldsymbol{x}'''(t_n)$ gives

$$\boldsymbol{x}'''(t_n) = \frac{3!(1+2\omega_n)(1+\omega_n)\big(\boldsymbol{x}^{\mathrm{c}}_{n+1} - \boldsymbol{x}^{\mathrm{f}}_{n+1}\big)}{(\frac{1}{2}\omega_n^2 + \frac{3}{4}\omega_n + \frac{1}{2})h_n^3 + (\omega_n^2 + \frac{4}{3}\omega_n + \frac{2}{3})h_n^2 h_{n-1}},$$

and substituting into (4.16) we obtain

$$\boldsymbol{\epsilon}_{\mathrm{c}} \approx -\frac{(1+\omega_n)^3}{\frac{1}{2}\omega_n^3 + \frac{7}{4}\omega_n^2 + \frac{11}{6}\omega_n + \frac{2}{3}} (\boldsymbol{x}^{\mathrm{c}}_{n+1} - \boldsymbol{x}^{\mathrm{f}}_{n+1}).$$

We are thus able to estimate the LTE of the coarse step without computing any derivatives using the coarse and fine approximations. We note that in this case $\boldsymbol{\epsilon}_{\mathrm{c}}$ is a vector and so we use the max-norm, $\epsilon_{\mathrm{c}} = \|\boldsymbol{\epsilon}_{\mathrm{c}}\|_\infty$, to determine whether or not the error lies within the acceptable range.

4.2.3 Richardson Extrapolation

If ϵ_{c} is within the acceptable range, then we can accept the coarse approximation $\boldsymbol{x}^{\mathrm{c}}_{n+1}$ as the next value. However, since we already have an approximation for $\boldsymbol{\epsilon}_{\mathrm{c}} = \boldsymbol{x}(t_{n+1}) - \boldsymbol{x}^{\mathrm{c}}_{n+1}$, we can use this to obtain a numerical scheme of one order higher (3 as opposed to 2 for us) by taking the following linear combination of $\boldsymbol{x}^{\mathrm{c}}_{n+1}$ and $\boldsymbol{x}^{\mathrm{f}}_{n+1}$

$$\boldsymbol{x}(t_{n+1}) = \boldsymbol{x}^{\mathrm{c}}_{n+1} + \boldsymbol{\epsilon}_{\mathrm{c}} \approx -\frac{\frac{1}{2}\omega_n^3 + \frac{5}{4}\omega_n^2 + \frac{7}{6}\omega_n + \frac{1}{3}}{\frac{1}{2}\omega_n^3 + \frac{7}{4}\omega_n^2 + \frac{11}{6}\omega_n + \frac{2}{3}} \boldsymbol{x}^{\mathrm{c}}_{n+1} + \frac{(1+\omega_n)^3}{\frac{1}{2}\omega_n^3 + \frac{7}{4}\omega_n^2 + \frac{11}{6}\omega_n + \frac{2}{3}} \boldsymbol{x}^{\mathrm{f}}_{n+1}.$$

Despite its reported stability issues [130], Richardson extrapolation has performed well for our retinal dynamics model.

4.2.4 Convergence Study

Since the underlying time-stepping scheme in our solver is BDF2, we expect to have a global error that is $O(\Delta t^2)$. We confirm this with a convergence analysis using numerical simulation 1 (see Appendix A), in which the light stimuli is constant in order to preserve the C^3 requirement needed for the Taylor series (see Section 4.2.2).

Since the true solution is not available for our numerical simulations, we use the approximation

$$err^{\Delta t} := \frac{\left\| \boldsymbol{V_m}^{\Delta t} - \boldsymbol{V_m}^{\Delta t^*} \right\|_\infty}{1 - \left(\frac{\Delta t^*}{\Delta t}\right)^p} \approx C\Delta t^p, \tag{4.17}$$

in which $\boldsymbol{V_m}^{\Delta t}$ is the solution $\boldsymbol{V_m}$ using constant step-size Δt, Δt^* is the finest constant step-size used to approximate the solution to numerical simulation 1, C is a constant, and p is the order of the method. The modification term $1 - \left(\frac{\Delta t^*}{\Delta t}\right)^p$ approaches unity as Δt^* vanishes since $\boldsymbol{V_m}^{\Delta t^*}$ becomes the true value, yielding the usual error approximation formula. Figure 4.3 shows the logarithmic relation between $err^{\Delta t}$ error and Δt. We obtain slope $p = 2$ confirming the expected order of our method.

4.2.5 Wall Time and Accuracy

We also use numerical simulation 1 to compare the wall time and accuracy of the adaptive time-stepper and the constant time-stepper. Figure 4.3 shows that the adaptive time-stepper, for most choices of *tol* (see Section 4.2.1), improves the accuracy by a factor of at least 10, as compared to the constant time-stepper with similar wall time. Figure 4.3 makes it clear that we can use the constant time-stepper with larger time-steps and achieve better wall time than that of the adaptive time-stepper, but the cost

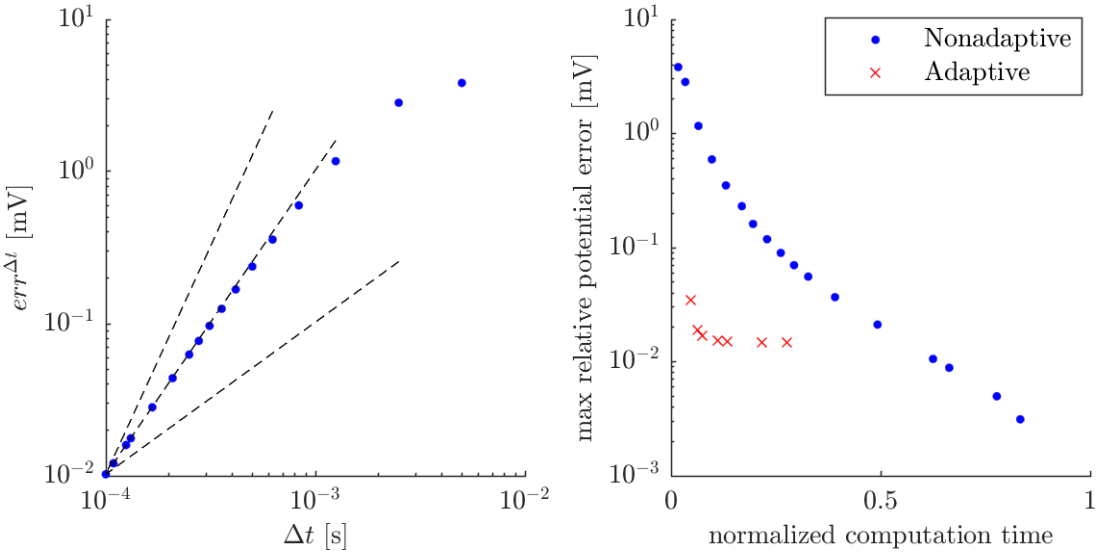


Figure 4.3: Left: A convergence study for our time-stepping method showing the relation between $err^{\Delta t}$ and the step-size. The dashed lines have slopes 1, 2, and 3. The expected order of 2 is observed. Right: The relation between the wall time and the maximum relative potential error (as compared to the finest constant simulation) for various adaptive (×), and constant time-step (•) simulations. Accuracy and simulation time are, roughly, inversely proportional. In some cases, the adaptive time-stepper decreases the error by a factor of around 100 as compared to a simulation of similar wall time. The difference between the various adaptive simulations is the *tol* value chosen (see Section 4.2.1). We ran adaptive simulations for $tol = 5 \cdot 10^{-3}, 10^{-3}, 5 \cdot 10^{-4}, 10^{-4}, 5 \cdot 10^{-5}, 10^{-5}, 5 \cdot 10^{-6}$ (see Fig. 4.7).

would be a loss in accuracy. Similarly, using the constant time-stepper with much smaller time-steps achieves better accuracy than that of the adaptive time-stepper, but the cost would be much slower simulations. So while the adaptive time-stepper is neither the most accurate nor the fastest simulation, it is the best compromise.

This result depends on the type of experiment conducted. For example, an experiment with a shorter light pulse would increase the efficiency of the adaptive time-stepper as it will use larger time-steps in the absence of light stimuli (see Section 4.2.6). On the other hand, an experiment with multiple light pulses would decrease the efficiency of the adaptive time-stepper as it would need to adjust the step-size numerous times. Such light stimuli would also affect the constant time-stepper as only small step-sizes would yield reasonably accurate results.

4.2.6 In-depth Analysis of the Adaptive Time-stepper

We use numerical simulation 2 (see Appendix A) to study the adaptive time-stepper in greater detail. Figure 4.4 offers a deeper look on how the adaptive time-stepper works. It shows how the step-size gradually increases until a suitable step-size is found. This suitable step-size varies depending on the situation, for example, in the presence of a light stimulus ($t \leq 0.02$ s), the step-size that the adaptive time-stepper stabilized on was much smaller than that in the absence of light ($t > 0.02$ s). This is expected as the photoreceptor dynamics are undergoing much more rapid change in the presence of light. Another feature shown in Fig. 4.4 is the gradual, rather than sudden, change of step-size. This is evident as for most time-steps at most one rejection was made.

Figure 4.4 also shows a few safety guards put in place for the adaptive time-stepper. These include a maximum and minimum allowed step-size, and forcing the time-stepper to step to certain critical time values. The reason for the latter is that when the light stimulus shuts off we lose differentiability and so using BDF2 would

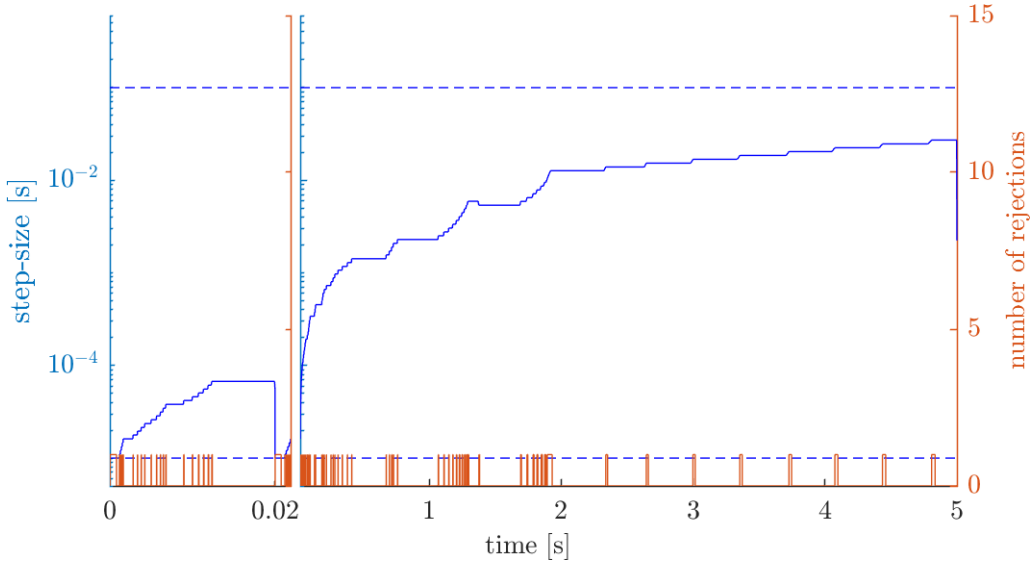


Figure 4.4: Step-size (left y-axis) and number of rejections (right y-axis) as functions of time. The dashed lines are the minimum and maximum allowed step-sizes, $\Delta t_{\min}$ and $\Delta t_{\max}$. Any decision of the adaptive stepper to take steps of size outside of the interval $[\Delta t_{\min}, \Delta t_{\max}]$ will be overridden by a safety feature. Around $t = 0.02$ s the step-size decreases sharply as we force the time-stepper to step to certain critical times such as the beginning and end of light stimuli. This is also observed for the step at $t = 5$ s. As the dynamics of the photoreceptors change less frequently (due to absence of any light stimuli) the adaptive stepper uses larger and larger time-steps.

not be justified. Instead, we step to the critical time and use the improved forward Euler method, whose order is the same as BDF2, to take the subsequent first step.

4.2.7 Comparison of Adaptive Time-steppers

Our work on the adaptive time-stepper was partially inspired by [130]. In this section we will compare the two time-steppers to highlight the key differences and compare the overall performances of the two on a simple problem. The two main differences are in computing the fine approximation, $\boldsymbol{x}^{\mathrm{f}}_{n+1}$, and the local error estimations. To better compare the local error estimations, we have implemented an adaptive time-stepper identical to that of [130], but with our local error estimates. In what follows, we will refer to that time-stepper as the memory-intensive B (MI-B) time-stepper, to the time-stepper in [130] as the memory-intensive M (MI-M) time-stepper, and to

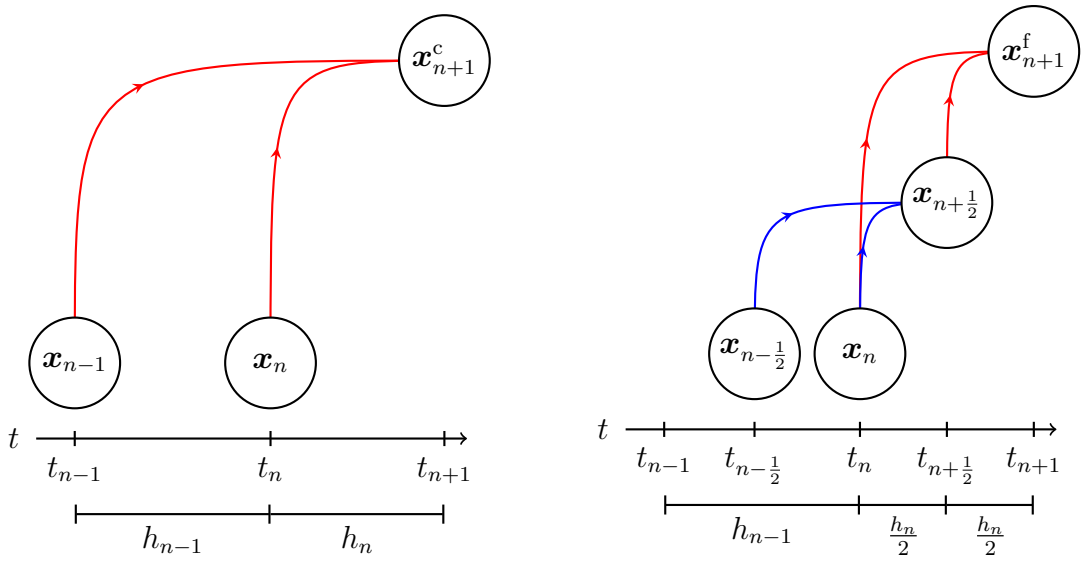


Figure 4.5: Schematic depiction for the MI time-steppers. Left: The coarse approximation of $\boldsymbol{x}$ at t_{n+1}, $\boldsymbol{x}^{\mathrm{c}}_{n+1}$, is computed identically to what was done in the ME time-stepper (see Fig. 4.2). Right: The fine approximation of $\boldsymbol{x}$ at t_{n+1}, $\boldsymbol{x}^{\mathrm{f}}_{n+1}$, is computed in two steps. First an intermediate value, $\boldsymbol{x}_{n+\frac{1}{2}}$, is computed using $\boldsymbol{x}_{n-\frac{1}{2}}$ and $\boldsymbol{x}_n$ (shown in blue). Then $\boldsymbol{x}^{\mathrm{f}}_{n+1}$ is computed using $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n+\frac{1}{2}}$ identically to what was done in the ME time-stepper. The difference in calculating the intermediate value in the fine approximation necessitates keeping an additional set of historical values, $\boldsymbol{x}_{n-\frac{1}{2}}$, for the MI time-stepper.

our original time-stepper as the memory-efficient (ME) time-stepper. Also, when the distinction is not necessary we will refer to either of the MI-B and MI-M time-steppers as the MI time-steppers.

Addressing the first difference, the ME time-stepper uses the same historical data, namely $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n-1}$, to compute both the coarse approximation of $\boldsymbol{x}$ and the approximation of $\boldsymbol{x}$ at the half-step, $\boldsymbol{x}_{n+\frac{1}{2}}$ (see Fig. 4.2). On the other hand, as is shown in Fig. 4.5, the MI time-steppers use $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n-\frac{1}{2}}$ for the approximation of $\boldsymbol{x}$ at the half-step, while the coarse approximation is still computed using $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n-1}$. This implies that the MI time-steppers must keep track of one more set of historical data, namely the previous half-step values, $\boldsymbol{x}_{n-\frac{1}{2}}$. For simple models this extra memory cost might be negligible, however, for complicated models, such as ours, the cost will be burdensome.

As for the difference in local error estimation, it mostly stems from missing the $\frac{1+\omega_n}{1+2\omega_n}$ factor in (4.15). This factor arises from using variable step-sizes as opposed to the constant step-size BDF2.

To confirm our LTE calculation and to study further ramifications of these differences, we study the scalar initial value problem

$$\frac{dx}{dt} = -x + g(t), \tag{4.18}$$

$$g(t) = \sin\left(t\left(1 + 5\exp\left(-\frac{(t-5)^2}{4}\right)\right)\right), \tag{4.19}$$

$$x(0) = -\frac{1}{2}, \tag{4.20}$$

over the time interval $[0, 30]$. Using integrating factors, we can determine that the exact solution to this initial value problem is $x(t) = e^{-t}\left(\int_0^t e^t g(t)dt - \frac{1}{2}\right)$. This allows us to compare the predicted coarse LTE, ϵ_c, to the actual coarse LTE, ϵ_c^* (see Fig. 4.6). We do this by using the adaptive time-steppers to step one time-step using accurate historical data from the known analytic solution. Figure 4.6 shows that for almost all time-steps, both the MI-B and ME time-steppers LTE predictions are two orders of magnitude more accurate than that of the MI-M time-stepper. This supports the validity of our error calculations, especially since this is the only difference between the two MI time-steppers.

All that said, we have observed that all time-steppers tested behaved roughly the same from a global view. Figure 4.6 shows global error of the various methods for the initial value problem at hand. All three time-steppers have approximately the same global error throughout the simulation. A possible explanation for this is that the LTE estimate in the MI-M time-stepper is an overestimate of that in the MI-B and ME methods, since $\left\|\frac{1+\omega_n}{1+2\omega_n}\right\| < 1$. This implies that the MI-M time-stepper will not be taking bigger time-steps due to the faulty error approximation. Hence, we do not expect the accumulation of any extra errors from this difference. Furthermore,

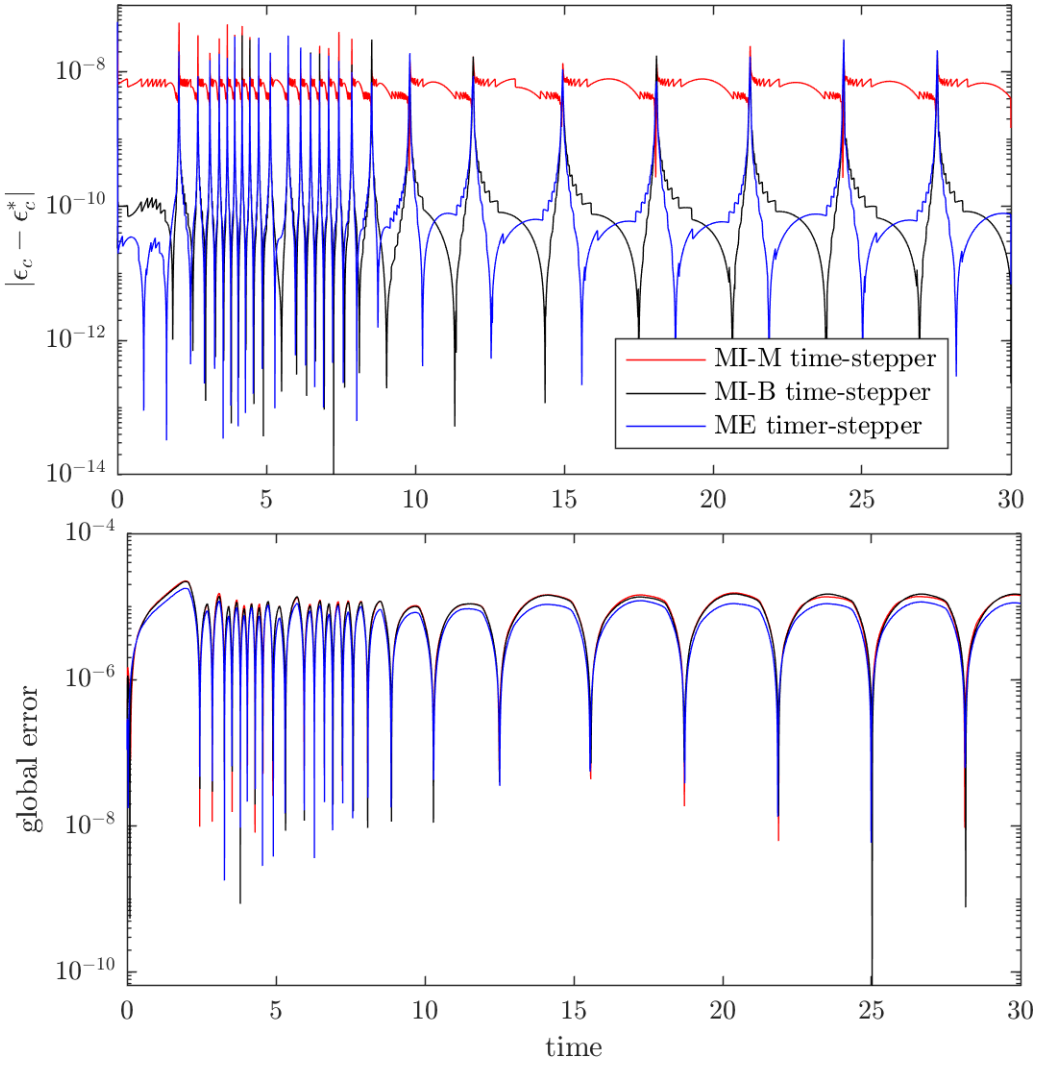


Figure 4.6: Top: The difference between the predicted coarse LTE, ϵ_c, and the actual coarse LTE, ϵ_c^*, of the various methods as a function of time. Using our error estimate yields a much more accurate LTE approximation as opposed to the standard error estimate. Bottom: Global error of the various methods as a function of time. All three methods roughly have the same global error.

an LTE is considered suitable if it falls within some interval containing our desired LTE. This along with the safety factors (see Section 4.2.1) decrease the effect of the differences in the error approximations on the computed solutions.

4.3 Solving the Linear Jacobian Update Equation

As discussed in Section 4.2, the linear Jacobian update equation can be written as

$$D^i \Delta \boldsymbol{x}^{i+1} = \boldsymbol{r}^i. \tag{4.21}$$

After ensuring uniqueness of the solution (see Section 2.2.2), D^i's are $(n+1) \times n$, large, sparse matrices. Solving these updates directly proved to be onerous and expensive. This is expected as the system Jacobian is quite large and complicated, mixing between diffusion of electrical potentials and chemical reactions occurring in photoreceptors. However, the Jacobian is sparse and highly structured (see Fig. 4.9), which can be exploited for a significant reduction in cost. Iterative methods are particularly well suited for such problems [119, 104]. Two iterative methods that we found success with in solving (4.21) were the Richardson-D'Jakonov iteration and GMRES. In this section, we discuss our use of these two methods, including our attempts to further reduce their cost.

4.3.1 Richardson-D'Jakonov Iteration

Using our terminology in (4.21), D'Jakonov's proposed iteration (3.3) is based on the splitting

$$\gamma_j D^i = B + (\gamma_j D^i - B), \tag{4.22}$$

in which γ_j may vary with each inner iteration. We choose $B = D^0$ and $\gamma_j = 1$. Hence, our inner iteration is a semi-explicit iteration given by

$$D^0 \Delta \boldsymbol{x}_{j+1}^{i+1} = \boldsymbol{r}^i - (D^i - D^0)\Delta \boldsymbol{x}_j^{i+1}, \tag{4.23}$$

in which $\Delta \boldsymbol{x}_j^{i+1}$ is a sequence that converges to $\Delta \boldsymbol{x}^{i+1}$ (we choose $\Delta \boldsymbol{x}_0^{i+1} = \mathbf{0}$).

As is common with iterative solvers, we choose to solve (4.21) only approximately. Once the relative residual,

$$\frac{\left\| D^i \Delta \boldsymbol{x}_j^{i+1} - \boldsymbol{r}^i \right\|_\infty}{\left\| \boldsymbol{r}^i \right\|_\infty},$$

decreases below 10^{-6}, we halt the iterative solver and accept the iterate $\Delta \boldsymbol{x}_j^{i+1}$. Hence, in this setting, we are solving (4.14) using an inexact Newton method [116, 31].

The motivation for the choice of splitting of D^i, seen in (4.22), is two fold. First, through our experimentation with this model, we notice that the Newton method converges very quickly (around 3 iterations) for most time-steps. This, the relatively small step-sizes the solver takes, and the nature of our model suggests that D^0 is an adequate approximation of D^i. Second, this choice allows us to decompose D^0 once and use the decomposition to solve all subsequent iterations (for each application of this Newton-iterative method). This makes solving (4.23) significantly faster in comparison to direct solves of (4.21). To illustrate, we use numerical simulation 1 to compare the performance of the direct and iterative solvers. Figure 4.7 shows that, given a specific *tol* value (see Section 4.2.1), the iterative solver is twice as fast as the direct solver while the accuracy of the two solvers are identical. This means that all the advantages of the adaptive-direct time-stepper are retained by the adaptive-iterative time-stepper. Recalling Fig. 4.3, this implies the adaptive-iterative time-stepper improves accuracy by a factor of at least 10, as compared to the constant time-stepper with twice the wall time.

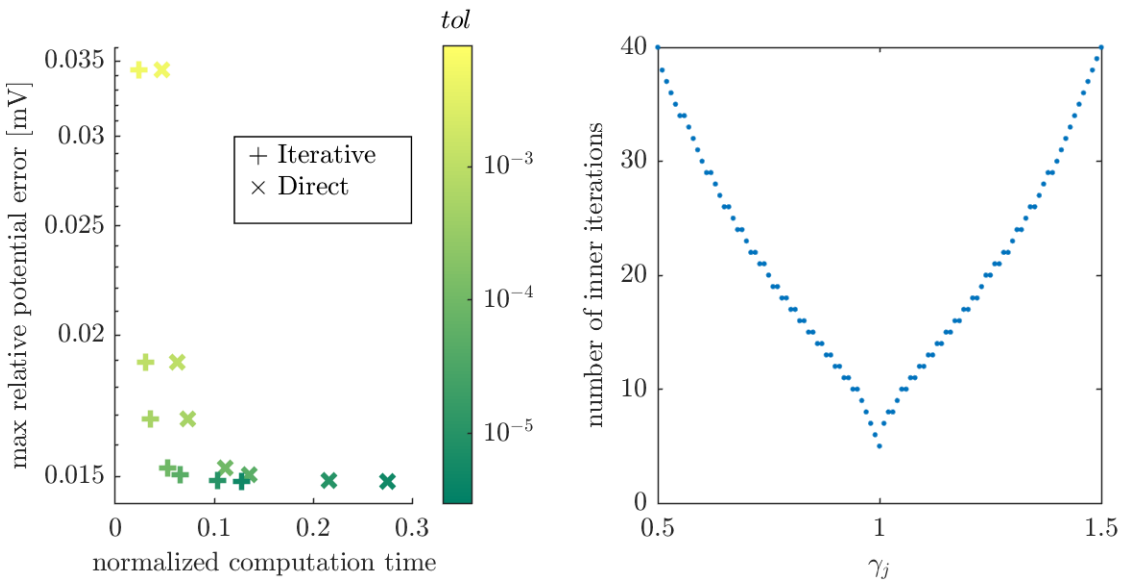


Figure 4.7: Left: The relation between the wall time and the maximum relative potential error (as compared to the finest constant simulation) for various adaptive-direct (×), and adaptive-iterative (+) simulations. The colour bar on the right indicates the *tol* values. For all *tol* values tested, the iterative solver improves the wall time of simulations by roughly a factor of 2 while retaining the same accuracy level (also shown in Fig. 4.3). Right: Number of inner iterations needed to achieve the desired relative residual level as a function of γ_j. The curve, although just an example, is basically the same through out our experimentation. In particular, in our experience $\gamma_j = 1$ has always been the optimal choice and the more γ_j deviates from unity the more inner iterations required to achieve the desired relative residual level.

The choice of $\gamma_j = 1$, in (4.22), is also based on the intuition that D^0 is an adequate approximation to D^i. We verified this by experimenting with the model, as shown in Fig. 4.7. Figure 4.7 shows the number of inner iterations needed to achieve the desired relative residual level as a function of γ_j. It is clear the $\gamma_j = 1$ is the optimal choice to minimize the number of inner iterations needed. We also observed that the larger $\|\gamma_j - 1\|$ gets the more inner iterations are required to achieve the desired relative residual level. These results hold true through out our extensive experience with the model.

In addition, the number of inner iterations is also affected by the adaptive time-stepper *tol* value (see Section 4.2.1). Figure 4.8 shows a positive correlation between *tol* values and number of inner iterations required. This was expected as lower *tol*

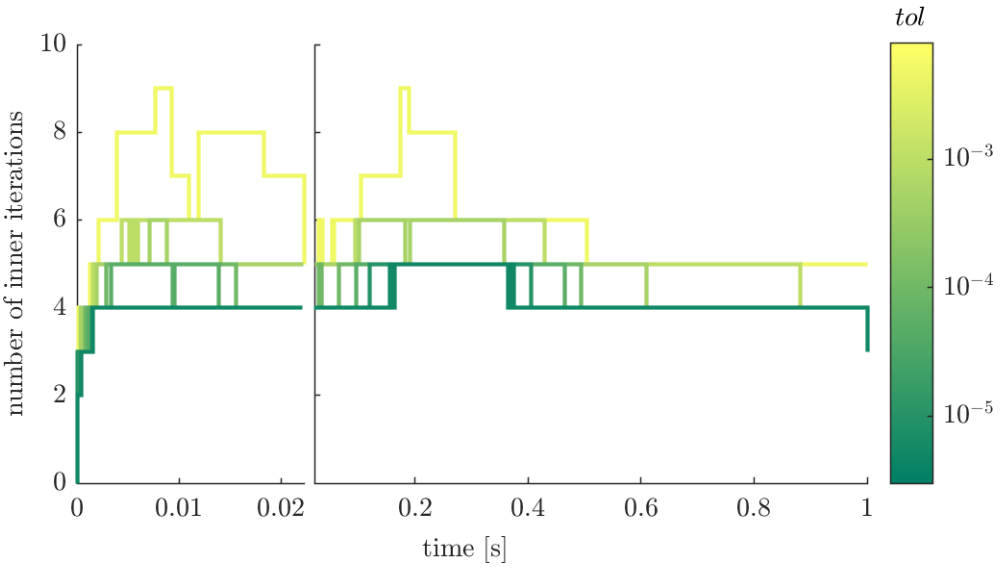


Figure 4.8: The number of inner iterations required (for all *tol* values tested) as a function of time. The colour bar on the right indicates the *tol* values. For all simulations the maximum number of inner iterations needed is less than 10. We observe that *tol* values and number of inner iterations required have a positive correlation. This is expected as simulations with lower *tol* values take smaller step-sizes and so the solution to the Jacobian update equation (4.21) is closer to the initial guess as compared to those with higher *tol* values.

values force the time-stepper to take smaller steps, which means the solution to the Jacobian update equation (4.21) is not far from the initial guess $\Delta \boldsymbol{x}_0^{i+1} = \mathbf{0}$. This allows the iterative solver to converge faster for smaller *tol* values .

4.3.2 Generalized Minimal Residual Method

Unlike the Richardson-D'Jakonov iteration, GMRES is not readily adaptable to non-square systems. Hence, we must forgo the uniqueness condition (2.7) and apply GMRES on the square, singular Jacobian matrix. For the remainder of this chapter, the Jacobian matrix, D^i, refers to the $n \times n$ system Jacobian which does not include the appended extra row for ensuring uniqueness (see Section 2.2.2). Two immediate concerns arise as a result of this choice. The first concern is we are applying GMRES to solve the Jacobian update equation (4.21), in which the matrix, D^i, is singular. Can GMRES find solutions to singular systems? The other concern is, suppose we are

able to obtain a solution to (4.21), how can we ensure that the solutions obtained at various time-steps correspond to the single solution of the PDE system (2.3) to (2.6) that we are interested in?

As for the first concern, a considerable amount of research has been directed at applying GMRES to singular and nearly singular systems, for example [108, 77, 19, 21, 18, 86, 96]. In fact, some of these studies offered insights as to when GMRES can be applied to find a solution (in the least square sense) of inconsistent systems [77, 19, 18]. As for the case of consistent but singular systems, such as ours, the issue of applying GMRES is that a suitable solution may not lie in the Krylov subspace (see Section 3.2) [60]. Fortunately, some progress have been made as to when singular consistent systems admit a Krylov solution [60, 18, 96]. In our case, we are guaranteed the existence of a Krylov solution since we can check that the range of D^i intersects its one dimensional null space trivially [18].

The second concern is easier to address. Assuming that, for a specific time-step, we are able to find a solution, $V_{\mathrm{m}}, \phi_{\mathrm{e}}, \phi_{\mathrm{s}}$ to the PDE system (2.3) to (2.6). Since, for any constant, $V_{\mathrm{m}}, \phi_{\mathrm{e}} + c, \phi_{\mathrm{s}} + c$ is also a solution (see Section 2.2.2), we can simply choose c so that (2.7) is satisfied. More concretely, we pick c to satisfy

$$\int_{\partial\mathcal{S}\setminus\partial\mathcal{R}} (\phi_{\mathrm{s}} + c) + \int_{\partial\mathcal{R}_{\mathrm{o}}} (\phi_{\mathrm{e}} + c) = 0.$$

Equivalently,

$$c = -\frac{1}{\mathrm{area}(\partial\mathcal{S})}\left(\int_{\partial\mathcal{S}\setminus\partial\mathcal{R}} \phi_{\mathrm{s}} + \int_{\partial\mathcal{R}_{\mathrm{o}}} \phi_{\mathrm{e}}\right),$$

in which $\mathrm{area}(\partial\mathcal{S}) = \displaystyle\int_{\partial\mathcal{S}} 1 = 4\pi r_{\mathrm{eye}}^2$ is the surface area of the eye. It is not difficult to see that by doing this at every time-step we end up at our desired solution (described in Section 2.2.2).

After resolving both concerns, we were immediately faced with another problem:

a straight forward application of GMRES on (4.21) was not fruitful. GMRES was not converging, if at all, after a reasonable amount of iteration, regardless of how we vary the restart parameter m in GMRES(m) (see Section 3.2.7). This was not completely unexpected since, as we mention in Section 3.3, preconditioners are essential to the success of iterative methods.

Most of our effective approaches centered around ILU preconditioners (see Section 3.3.1 for an introduction). Specifically, ILU(0) and the more sophisticated ILUT preconditioners. Unsurprisingly, the latter of the two is significantly improved when combined with a reordering and rescaling preconditioner (see Section 3.3.2 for an introduction). We discuss the various permutations and the scaling of the Jacobian matrices before comparing the overall performances.

Jacobian Reorderings

We begin by addressing the initial ordering of the system Jacobian. We arbitrarily choose the order of the equations to be (4.6) to (4.8), respectively. Noting the dependency of these equations on the variables (see Fig. 4.9), it is clear that the natural order of the variables should start with $\boldsymbol{\phi}_\mathbf{s}$ and $\boldsymbol{\phi}_\mathbf{e}$, respectively, and then $\boldsymbol{V}_\mathbf{m}$ and the auxiliary variables $\boldsymbol{X}$. Otherwise, most nonzero entries will be away from the diagonal, which will make solving (4.21) more difficult.

Figure 4.9 shows the sparsity pattern of the Jacobian with this initial ordering, which will be referred to as the standard ordering. It is evident that the Jacobian is very sparse, with roughly 0.003% of its entries being nonzero, and highly structured. One can readily see the dependency of (4.6) on $\boldsymbol{\phi}_\mathbf{s}$ and $\boldsymbol{\phi}_\mathbf{e}$ from the top block; (4.7) on $\boldsymbol{\phi}_\mathbf{s}$, $\boldsymbol{\phi}_\mathbf{e}$, and $\boldsymbol{V}_\mathbf{m}$ from the middle block; and (4.8) on all the variables from the bottom block. The bottom block also shows the intercellular interactions, which manifest as the off-diagonal bands in the bottom-right block of the Jacobian. Intracellular interactions also appear in the bottom-right block of the Jacobian as the diagonal

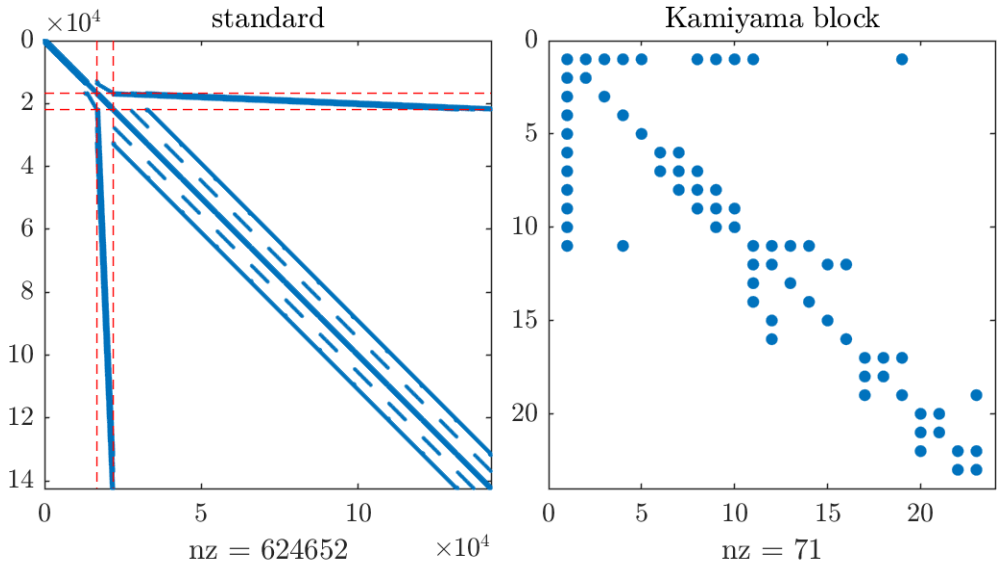


Figure 4.9: Left: Sparsity pattern of the Jacobian matrix in the standard ordering. The horizontal dashed lines segment the matrix into the three equations (4.6) to (4.8), respectively. The vertical dashed lines segment the variables into $\boldsymbol{\phi}_\mathbf{s}$, $\boldsymbol{\phi}_\mathbf{e}$, and $\boldsymbol{V}_\mathbf{m}$ and the auxiliary variables $\boldsymbol{X}$ (ordered per discretization node), respectively. The Jacobian matrix is very sparse (only about 0.003% of its entries are nonzero) and highly structured. Broad bandwidth and upward-pointing arrow submatrices are clearly observed. Right: Sparsity pattern of the Kamiyama block submatrix, in which $\boldsymbol{V}_\mathbf{m}$ correspond to the first column. The dependency pattern clearly shows upward-pointing arrow submatrices as well.

blocks (shown in Fig. 4.9), which correspond to $\boldsymbol{V}_\mathbf{m}$ (first column) and $\boldsymbol{X}$, associated to each discretization node (hereafter referred to as the Kamiyama block).

From Fig. 4.9 it is evident that the Jacobian, in its current ordering, has a very broad bandwidth. Indeed, the dependency of the middle block on $\boldsymbol{V}_\mathbf{m}$ and the bottom block on $\boldsymbol{\phi}_\mathbf{s}$ and $\boldsymbol{\phi}_\mathbf{e}$ is the main cause of this undesirable property. However, upon examining the PDE system (2.3) to (2.6), it is clear that only the $\boldsymbol{V}_\mathbf{m}$ equations (corresponding to (2.5)) in (4.8) that depends on $\boldsymbol{\phi}_\mathbf{s}$ and $\boldsymbol{\phi}_\mathbf{e}$. Since the middle block only depends on $\boldsymbol{V}_\mathbf{m}$ and not $\boldsymbol{X}$, (symmetrically) reordering the Jacobian so that $\boldsymbol{V}_\mathbf{m}$ for all the discretization nodes appear first and then $\boldsymbol{X}$ may significantly reduce the bandwidth. This ordering will be referred to as the $\boldsymbol{V}_\mathbf{m}$-first ordering and is shown in Fig. 4.10. We note that this ordering will also make the aforementioned off-diagonal entries in the bottom-right block, corresponding to the intercellular inter-

actions disappear (see Fig. 4.10). This is because all the direct intercellular interaction occur between the $\boldsymbol{V}_{\mathbf{m}}$ variables. The auxiliary variables $\boldsymbol{X}$ are directly affected only through intracellular interactions.

An additional apparent feature of the Jacobian in Fig. 4.9 is that the arrow submatrices are pointing up. This is noticeable in the Kamiyama blocks and the larger arrow submatrices forming as a result of the dependency of the top and middle blocks on $\boldsymbol{\phi}_{\mathbf{e}}$ and $\boldsymbol{\phi}_{\mathbf{s}}$, respectively, as well as the middle and bottom block on $\boldsymbol{V}_{\mathbf{m}}$ and $\boldsymbol{\phi}_{\mathbf{s}}$ and $\boldsymbol{\phi}_{\mathbf{e}}$, respectively. As discussed in Section 3.3.2, having these arrow submatrices point down is more numerically desirable for solving (4.21). This suggests that reversing the ordering of the Jacobian is a good reordering to consider. The reordering arising from reversing the standard ordering will, naturally, be referred to as the reverse standard ordering. It is shown in Fig. 4.10. We will also consider reversing the $\boldsymbol{V}_{\mathbf{m}}$-first ordering since the preceding argument applies to it as well. Naturally, that ordering will be referred to as the reverse $\boldsymbol{V}_{\mathbf{m}}$-first ordering and it is shown in Fig. 4.10.

In addition to the aforementioned orderings, we will also consider the RCM ordering. Hence, we will have five different orderings to consider; the Jacobian structure under the standard ordering is shown in Fig. 4.9, and the rest are shown in Fig. 4.10. The reason for considering the RCM ordering of the Jacobian is that studies have found it to improve the performance of ILU preconditioners [15, 14]. In fact, one can say that it became "common knowledge" to try the RCM ordering when applying ILU preconditioners since it is usually effective [104].

We gain further insight into these permutations by examining their corresponding Jacobian structure (shown in Fig. 4.10). A remarkable difference in the $\boldsymbol{V}_{\mathbf{m}}$-first ordering as compared to the standard ordering is absence of the off-diagonal bands in the bottom-right block. This was expected and may improve solving (4.21) as the large lower block has significantly less off-diagonal nonzero entries. However, the bandwidth of the Jacobian in the $\boldsymbol{V}_{\mathbf{m}}$-first ordering is only slightly reduced. This is

because the auxiliary variables $\boldsymbol{X}$ depend on $\boldsymbol{V_m}$, which is now ordered before $\boldsymbol{X}$ as opposed to being intertwined with it per discretization node. Clearly, both these points hold for the reverse orderings as well. We also observe, as we expected, that the arrow submatrices point downwards for the reverse orderings as opposed to upwards for the 'forward' ones. This may improve the ILU preconditioner as it reduces fill-in. The most significant bandwidth reduction is attained using the RCM ordering. This may also improve the ILU preconditioner as, on average, all nonzero entries are clustered near the diagonal compared to the other orderings.

These observations are optimistic in theory, however, it is not necessary that they translate into practical improvements. Before presenting our findings, we discuss the scaling we used since we found the most success when combining it with reordering.

Jacobian Scaling

In general, the convergence of Krylov methods depends on the eigenvalues of the system in question [104, 60]. In fact, the convergence of GMRES has been shown to depend on the radii and number of the eigenvalue clusters [20]. In the case where multiple eigenvalue clusters are present, GMRES may behave as if the linear system has one big eigenvalue cluster [20]. This along with Gershgorin's theorem indicate that a scaling that produces ones on the diagonals of the Jacobian and small off-diagonal entries may be effective.

Our Jacobian scaling method consists of iteratively normalizing the rows, under the max-norm, and then scaling the columns to have unit diagonal entires. Unambiguously, given a Jacobian matrix, D^i, and a preset number of scaling iteration, n, the new scaled matrix is RD^iC, in which $R = R_n \cdots R_1$ and $C = C_1 \cdots C_n$ are products of diagonal matrices. We define R_1 and C_1 to be the diagonal matrices with entries $(R_1)_{kk} = \frac{1}{\max_j\{(D^i)_{kj}\}}$ and $(C_1)_{kk} = \frac{\max_j\{(D^i)_{kj}\}}{(D^i)_{kk}}$, respectively. R_m and C_m are defined identically with $R_{m-1} \cdots R_1 D^i C_1 \cdots C_{m-1}$ in place of D^i. In simpler terms

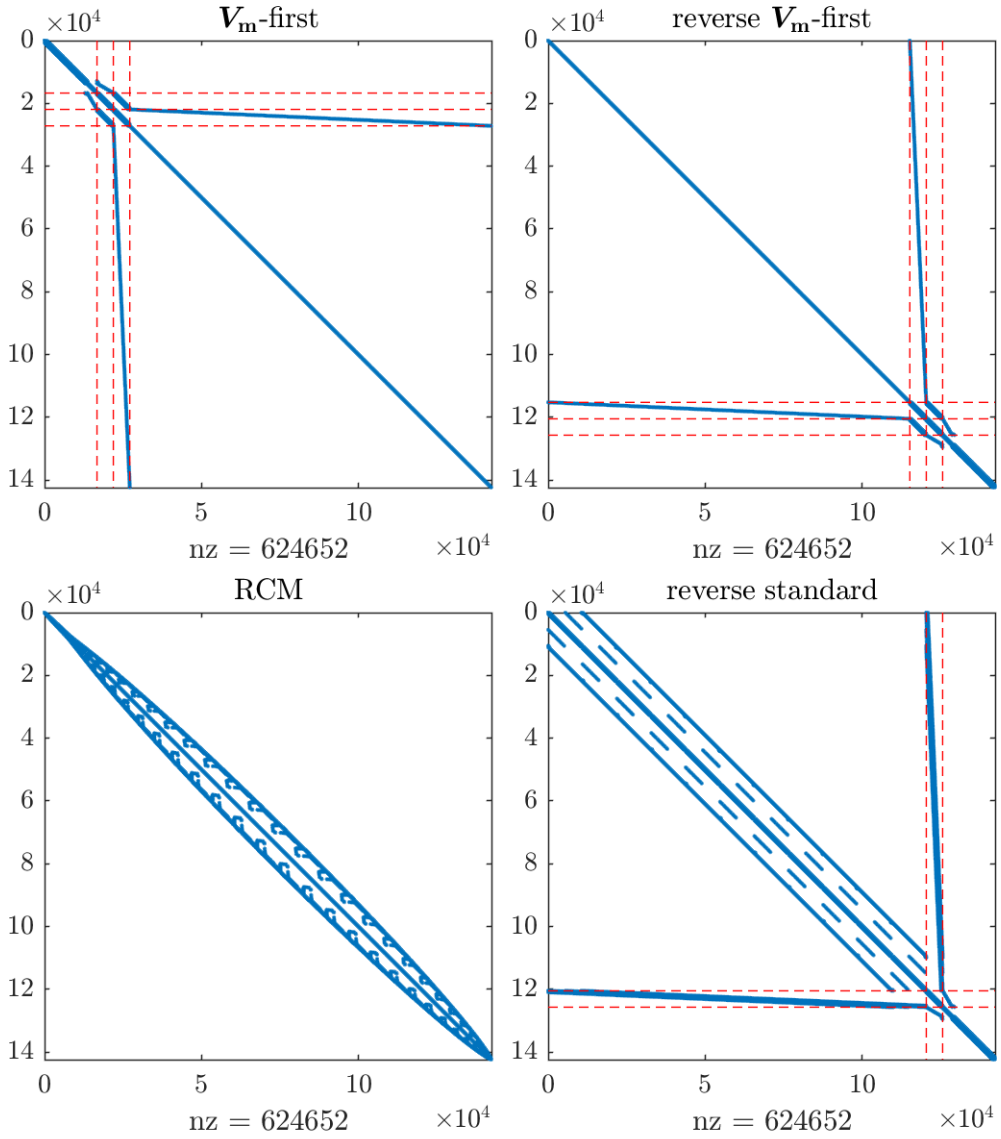


Figure 4.10: Top-left: Sparsity pattern of the Jacobian matrix in the $\boldsymbol{V}_\mathbf{m}$-first ordering. The horizontal dashed lines segment the matrix into (4.6), (4.7), the $\boldsymbol{V}_\mathbf{m}$ equations of (4.8), and the $\boldsymbol{X}$ equations of (4.8), respectively. The vertical dashed lines segment the variables into $\boldsymbol{\phi}_\mathbf{s}$, $\boldsymbol{\phi}_\mathbf{e}$, $\boldsymbol{V}_\mathbf{m}$, and $\boldsymbol{X}$, respectively. Top-right: Sparsity pattern of the Jacobian matrix in the reverse $\boldsymbol{V}_\mathbf{m}$-first ordering. The dashed lines segment the matrix in the exact opposite order as is done for $\boldsymbol{V}_\mathbf{m}$-first ordering (top-left). Bottom-left: Sparsity pattern of the Jacobian matrix in the RCM ordering. The RCM ordering exhibits the smallest bandwidth of all orderings, however, the physical intuition behind the ordering of the equations and variables is lost. The superior bandwidth reduction is not surprising considering the RCM ordering was designed for this purpose (as we mention in Section 3.3.2). Bottom-right: Sparsity pattern of the Jacobian matrix in the reverse standard ordering. The dashed lines segment the matrix in the exact opposite order as is done for the standard ordering (Fig. 4.9).

the scaling matrices are iteratively defined so that at each stage R_m normalizes the rows and C_m is chosen so that the diagonal entries are unity.

Theoretically the point of scaling the Jacobian more than once may not be clear, however, our experimental evidence shows it to be quite effective. We will refer to the time required to compute the ILU preconditioner of the system Jacobian as ILU computation time. Figure 4.11 shows a plot of the relation between the ILU computation time and the step-size that the adaptive time-stepper is taking (for various number of scaling iterations). The plot clearly shows that scaling once significantly reduces the ILU computation time. However, as the time-stepper takes bigger time-steps the ILU computation time significantly increases. Examining the number of rows of the Jacobian in which the diagonal entry is the largest entry (in absolute value) provides some justification to this observation. We will refer to such rows as diagonally dominant rows, Figure 4.11 shows a plot of the relation between the fraction of diagonally dominant rows of the system Jacobian and the step-size taken by the adaptive time-stepper (for various number of scaling iterations). It is clear that with bigger step-sizes less rows are diagonally dominant. This indicates that the overall Jacobian matrix is less diagonally dominant, which may be causing the increase in the ILU computation time.

We note that the results in Fig. 4.11 are not typical of all the orderings previously discussed. The results are representative only of the reverse orderings. We choose to limit our discussion to the reverse orderings here since, after considering all possible combinations, we obtained the best results with these orderings.

Also, as we mentioned in Section 3.3, preconditioning is problem dependent and this scaling technique is not an exception. In fact, similar scaling techniques have been found to perform poorly in some cases [27]. While more sophisticated (unsymmetric) reordering and scaling techniques have been developed [92, 37, 36], it is our experience that such techniques, for the problem at hand, performed comparably with

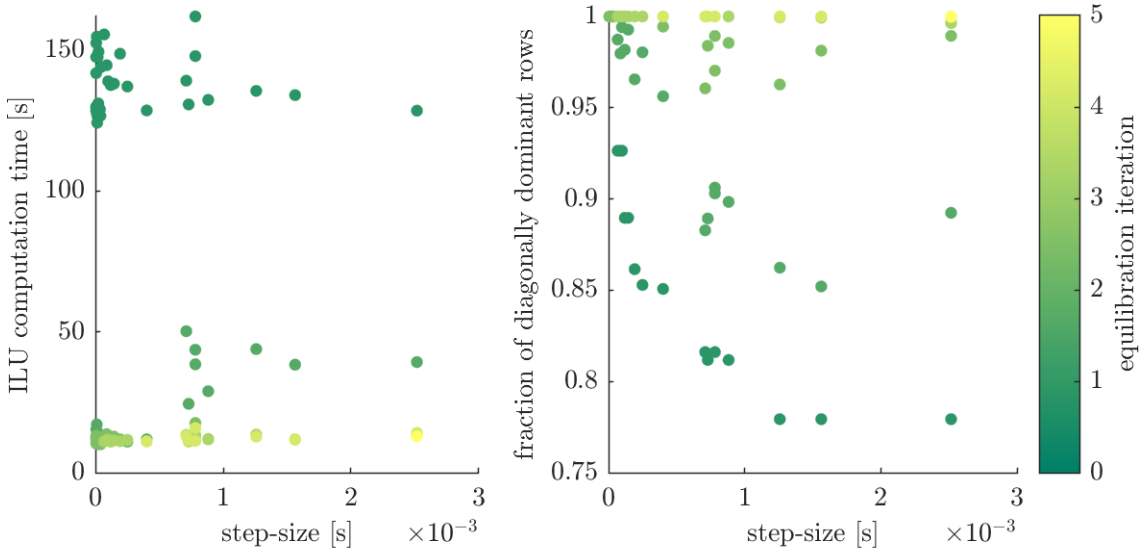


Figure 4.11: Left: A plot of the relation between ILU computation time [s] and the step-size [s] that the adaptive time-stepper decided to take for the various number of scaling iterations. Applying one scaling iteration significantly reduces ILU computation time, however, there is a positive correlation between ILU computation time and the step-size taken by the time-stepper. This correlation is not observed if more than one scaling iteration is applied. Right: A plot of the relation of the fraction of diagonally dominant rows and the step-size [s] that the adaptive time-stepper decided to take for the various number of scaling iterations. A strong negative correlation between the fraction of diagonally dominant rows and the step-size taken by the time-stepper is observed only when fewer than two scaling iterations are applied.

Table 4.1: Time [s] required for a Newton iteration for each of the preconditioners and for the various Jacobian orderings.

	Preconditioner		
Ordering	ILU(0)	ILUT	Delayed ILUT
Standard	2.5	37	14
Reverse Standard	2.5	12	4.7
$\boldsymbol{V}_{\mathbf{m}}$-first	2.5	37	14
Reverse $\boldsymbol{V}_{\mathbf{m}}$-first	2.5	12	4.7
RCM	2.5	19	7

the aforementioned method but were more costly.

Summary

We found more success when combining reordering with rescaling. More precisely, rescaling reduced the computation time by a factor of 2/3 (when comparing the best performing reorderings). For this reason, and to limit the number of combinations possible, we will only focus on ILU(0) and ILUT with (2 iterations of) scaling. We will also be considering a delayed version of the ILUT preconditioner. That is, we compute the ILUT preconditioner once for the first iteration of the Newton method and reuse the same one for subsequent iterations. We found great success with this method as it allows for the cost of computing the ILUT preconditioner to be split across the Newton iterations. Table 4.1 shows the time it takes for a single solve of (4.21) for the various preconditioners and orderings.

We found ILU(0) to be quite fast and is independent of the ordering (and, in fact, scaling). The time to compute the ILU(0) preconditioner was about 0.5 s with GMRES accounting for the majority of the Newton iteration time. However, the residual requirement in the Newton method needed to be relaxed for it to converge under this preconditioner. This did not seem to significantly affect the accuracy of the solution. In fact, all three preconditioners computed the solution with similar accuracy.

Unsurprisingly, the delayed ILUT preconditioner was more cost effective than the standard ILUT preconditioner. It is important to note that delaying the ILUT preconditioner had no effect on the convergence of either GMRES or the Newton method. The ILUT preconditioner was found to perform best on either of the reverse orderings. The superiority to the 'forward' orderings was somewhat expected as we discussed previously. ILUT with the RCM ordering performed well but not the best. However, we did observe it to be the most consistent regardless of scaling.

In summary, when comparing to the Richardson-D'Jakonov iteration, which takes about 3.5 s per Newton iteration, GMRES offers a faster alternative in ILU(0). However, as mentioned, this comes at the cost of relaxing the residual requirement in the Newton method. On the other hand, ILUT performs at least as well as the Richardson-D'Jakonov iteration but is slower. It also boasts the backing of theoretical and experimental results (as mentioned in Section 4.3.2), unlike our applications of the Richardson-D'Jakonov iteration to rectangular systems.

Chapter 5

Adaptive Time-Stepper Generalization

In this chapter, we generalize our adaptive time-stepper, that is based on variable step-size BDF2 (see Section 4.2) to variable step-size backward differentiation formulas (BDFs) of higher order. Our discussion will be mainly focused on BDFk, for $k = 3, \ldots, 6$. We omit BDF1 from our discussion since it is a one-step method. We also omit BDFk for $k > 6$ as they are unstable, even for constant step-sizes [53, 9, 54].

We recall that the BDFs are used in solving the differential equation $x'(t) = f(t, x)$ given by

$$k = 2: \; x_{n+1} + \alpha_0 x_n + \alpha_1 x_{n-1} = h_n \beta f(t_{n+1}, x_{n+1}), \tag{5.1}$$

$$k = 3: \; x_{n+1} + \alpha_0 x_n + \alpha_1 x_{n-1} + \alpha_2 x_{n-2} = h_n \beta f(t_{n+1}, x_{n+1}), \tag{5.2}$$

$$k = 4: \; x_{n+1} + \alpha_0 x_n + \alpha_1 x_{n-1} + \alpha_2 x_{n-2} + \alpha_3 x_{n-3} = h_n \beta f(t_{n+1}, x_{n+1}), \tag{5.3}$$

$$k = 5: \; x_{n+1} + \alpha_0 x_n + \alpha_1 x_{n-1} + \alpha_2 x_{n-2} + \alpha_3 x_{n-3} + \alpha_4 x_{n-4} = h_n \beta f(t_{n+1}, x_{n+1}), \tag{5.4}$$

$$k = 6: \; x_{n+1} + \alpha_0 x_n + \alpha_1 x_{n-1} + \alpha_2 x_{n-2} + \alpha_3 x_{n-3} + \alpha_4 x_{n-4} + \alpha_5 x_{n-5} = h_n \beta f(t_{n+1}, x_{n+1}), \tag{5.5}$$

in which $x_n \approx x(t_n)$, $t_{n+1} = t_n + h_n$, and α_i and β are coefficients chosen so that BDFk will have order k. We start by deriving the α_i and β coefficients, and the linear systems that determine them, for all the BDFk for $k = 2, \ldots, 6$. We then move on to discuss how our time-stepping scheme generalizes to the rest of the BDFs and derive the required error estimates. We conclude this chapter with an example to compare the adaptive time-stepper of various orders.

5.1 Determining the BDF Coefficients

The standard way of determining the BDF coefficients is by using Taylor series on the local truncation error LTE $= x(t_{n+1}) - x_{n+1}$, in which x_{n+1} is taken from the corresponding BDFk equation, one of (5.1) to (5.5). Subsequently setting the constants associated with $O(h^i), i \leq k$ terms to zero gives a system of equations (henceforth referred to as the BDFk conditions) that, once solved, gives the BDF coefficients. This approach is relatively easy for the constant step-size with some good references demonstrating it, see for example [9, 53]. However, clear references for deriving the variable step-size coefficients are hard to find. For this reason we will be detailed in our approach to this derivation, providing summary tables (see for example Table 5.1) where beneficial, as to serve as a good reference on this topic. We will, however, avoid repetition where possible as we hope to strike a balance between detailedness and readability.

5.1.1 Deriving the BDF Conditions

Since it has already been established that BDFk has LTE of order $k + 1$, our goal here is not to derive the sufficient conditions for the BDFs to satisfy this property. Instead, it suffices to obtain a system of necessary conditions for this requirement, that so happens to fully determine the BDF coefficients. This offers a much easier to

follow derivation of the BDF conditions as we hope to demonstrate.

The essential insight to this approach is BDFk must be able to determine polynomials of degree k or less with no error. Specifically, we are interested in the monomials $1, t, t^2, t^3, \ldots, t^k$ as they simplify computations considerably.

We start with the constant polynomial $x(t) = 1$. Since this is a degree 0 polynomial it must satisfy (5.1) to (5.5). In the case of BDF2, substituting $x_i = 1$ for all $i = n-1, n, n+1$ and $f(t_{n+1}, x_{n+1}) = x'(t_{n+1}) = 0$ into (5.1), we get

$$1 + \alpha_0 + \alpha_1 = 0 \tag{5.6}$$

must hold. A summary of all the BDF conditions is given in Table 5.1. Similarly, for BDF3 we get

$$1 + \alpha_0 + \alpha_1 + \alpha_2 = 0 \tag{5.7}$$

and the pattern is obvious for the rest of BDFs (see Table 5.1).

Also, the linear polynomial $x(t) = t$ must satisfy (5.1) to (5.5). For BDF2, using $x_n = t_n$, $x_{n+1} = t_n + h_n = x_n + h_n$, $x_{n-1} = x_n - h_{n-1}$ and $f(t_{n+1}, x_{n+1}) = 1$, we get

$$(1 + \alpha_0 + \alpha_1)x_n + (h_n - \alpha_1 h_{n-1}) = h_n \beta.$$

Using (5.6) and rearranging gives us our second BDF2 condition

$$(1 - \beta)h_n - \alpha_1 h_{n-1} = 0. \tag{5.8}$$

Noting $x_{n-2} = x_n - (h_{n-1} + h_{n-2})$, the above can similarly be done to BDF3 giving its second condition

$$(1 - \beta)h_n - \alpha_1 h_{n-1} - \alpha_2(h_{n-1} + h_{n-2}) = 0. \tag{5.9}$$

The pattern continues in the same way for the rest of the BDFs and is included in Table 5.1.

Furthermore, the quadratic polynomial $x(t) = t^2$ must satisfy (5.1) to (5.5). For BDF2, using $x_{n-1} = (t_n - h_{n-1})^2 = x_n + 2h_n t_n + h_n^2$, $x_{n+1} = (t_n + h_n)^2 = x_n - 2h_{n-1}t_n + h_{n-1}^2$ and $f(t_{n+1}, x_{n+1}) = 2t_{n+1}$, we get

$$(1 + \alpha_0 + \alpha_1)x_n + 2(h_n - \alpha_1 h_{n-1})t_n + \left(h_n^2 + \alpha_1 h_{n-1}^2\right) = 2h_n t_{n+1}.$$

Using $t_{n+1} = t_n + h_n$ and rearranging, we get

$$(1 + \alpha_0 + \alpha_1)x_n + 2((1 - \beta)h_n - \alpha_1 h_{n-1})t_n + \left((1 - 2\beta)h_n^2 + \alpha_1 h_{n-1}^2\right) = 0.$$

which, after using (5.6) and (5.8) reduces to the BDF2 condition

$$(1 - 2\beta)h_n^2 + \alpha_1 h_{n-1}^2 = 0. \tag{5.10}$$

Equations (5.6), (5.8) and (5.10) fully determine the coefficients of BDF2 as we show in Section 5.1.2. The following procedure can similarly be followed to give the BDF3 counterpart

$$\begin{aligned}
&(t_n + h_n)^2 + \alpha_0 x_n + \alpha_1 (t_n - h_{n-1})^2 + \alpha_2 (t_n - (h_{n-1} + h_{n-2}))^2 = 2h_n \beta (t_n + h_n) \\
&(1 + \alpha_0 + \alpha_1 + \alpha_2)x_n + 2((1 - \beta)h_n - \alpha_1 h_{n-1} - \alpha_2 (h_{n-1} + h_{n-2}))t_n \\
&\qquad + \left((1 - 2\beta)h_n^2 + \alpha_1 h_{n-1}^2 + \alpha_2 (h_{n-1} + h_{n-2})^2\right) = 0, \\
&(1 - 2\beta)h_n^2 + \alpha_1 h_{n-1}^2 + \alpha_2 (h_{n-1} + h_{n-2})^2 = 0.
\end{aligned} \tag{5.11}$$

Unlike the previous polynomials, the polynomial $x(t) = t^3$ must only satisfy (5.2) to (5.5) since it is of degree 3. This is expected and inconsequential as the BDF2 coefficients are fully determined by (5.6), (5.8) and (5.10) as we already mentioned.

So here we start with BDF3; noting that $x_{n+1} = x_n + 3t_n^2 h_n + 3t_n h_n^2 + h_n^3$ we get

$$\begin{aligned}(1+\alpha_0+\alpha_1+\alpha_2)x_n + 3((1-\beta)h_n - \alpha_1 h_{n-1} - \alpha_2(h_{n-1}+h_{n-2}))t_n^2 \\ + 3\big((1-2\beta)h_n^2 + \alpha_1 h_{n-1}^2 + \alpha_2(h_{n-1}+h_{n-2})^2\big)t_n \\ + \big((1-3\beta)h_n^3 - \alpha_1 h_{n-1}^3 - \alpha_2(h_{n-1}+h_{n-2})^3\big) = 0.\end{aligned}$$

Applying conditions (5.7), (5.9) and (5.11) we get

$$(1-3\beta)h_n^3 - \alpha_1 h_{n-1}^3 - \alpha_2(h_{n-1}+h_{n-2})^3 = 0, \tag{5.12}$$

which completes all the BDF4 conditions, which fully determine the coefficients of BDF4.

We leave deriving the rest of the conditions to interested readers. It is not difficult, rather, it is tedious as the formulas get longer. A good strategy is to use the binomial expansion theorem (or Pascal's triangle), since it makes it easy to see how the calculations unfold. For example, the most cumbersome condition to derive is that for BDF6, which comes from letting $x(t) = t^6$. Since, in that case, $x_{n+1} = x_n^6 + 6t_n^5 h_n + 15t_n^4 h_n^2 + 20t_n^3 h_n^3 + 15t_n^2 h_n^4 + 6t_n h_n^5 + h_n^6$ and so plugging $x(t)$ into (5.5) gives

$$\begin{aligned}&\Big(1+\alpha_0+\alpha_1+\alpha_2+\alpha_3+\alpha_4+\alpha_5\Big)x_n \\ &\quad + 6\Big((1-\beta)h_n - \alpha_1 h_{n-1} - \alpha_2(h_{n-1}+h_{n-2}) - \alpha_3(h_{n-1}+h_{n-2}+h_{n-3}) \\ &\quad - \alpha_4(h_{n-1}+h_{n-2}+h_{n-3}+h_{n-4}) - \alpha_5(h_{n-1}+h_{n-2}+h_{n-3}+h_{n-4}+h_{n-5})\Big)t_n^5 \\ &\quad + 15\Big((1-2\beta)h_n^2 + \alpha_1 h_{n-1}^2 + \alpha_2(h_{n-1}+h_{n-2})^2 + \alpha_3(h_{n-1}+h_{n-2}+h_{n-3})^2 \\ &\quad + \alpha_4(h_{n-1}+h_{n-2}+h_{n-3}+h_{n-4})^2 + \alpha_5(h_{n-1}+h_{n-2}+h_{n-3}+h_{n-4}+h_{n-5})^2\Big)t_n^4\end{aligned}$$

$$
\begin{aligned}
&+ 20\Big((1-3\beta)h_n^3 - \alpha_1 h_{n-1}^3 - \alpha_2(h_{n-1}+h_{n-2})^3 - \alpha_3(h_{n-1}+h_{n-2}+h_{n-3})^3 \\
&- \alpha_4(h_{n-1}+h_{n-2}+h_{n-3}+h_{n-4})^3 - \alpha_5(h_{n-1}+h_{n-2}+h_{n-3}+h_{n-4}+h_{n-5})^3\Big)t_n^3 \\
&+ 15\Big((1-4\beta)h_n^4 + \alpha_1 h_{n-1}^4 + \alpha_2(h_{n-1}+h_{n-2})^4 + \alpha_3(h_{n-1}+h_{n-2}+h_{n-3})^4 \\
&+ \alpha_4(h_{n-1}+h_{n-2}+h_{n-3}+h_{n-4})^4 + \alpha_5(h_{n-1}+h_{n-2}+h_{n-3}+h_{n-4}+h_{n-5})^4\Big)t_n^2 \\
&+ 6\Big((1-5\beta)h_n^5 - \alpha_1 h_{n-1}^5 - \alpha_2(h_{n-1}+h_{n-2})^5 - \alpha_3(h_{n-1}+h_{n-2}+h_{n-3})^5 \\
&- \alpha_4(h_{n-1}+h_{n-2}+h_{n-3}+h_{n-4})^5 - \alpha_5(h_{n-1}+h_{n-2}+h_{n-3}+h_{n-4}+h_{n-5})^5\Big)t_n \\
&+ \Big((1-6\beta)h_n^6 + \alpha_1 h_{n-1}^6 + \alpha_2(h_{n-1}+h_{n-2})^6 + \alpha_3(h_{n-1}+h_{n-2}+h_{n-3})^6 \\
&+ \alpha_4(h_{n-1}+h_{n-2}+h_{n-3}+h_{n-4})^6 + \alpha_5(h_{n-1}+h_{n-2}+h_{n-3}+h_{n-4}+h_{n-5})^6\Big) = 0.
\end{aligned}
$$

We know all the terms except the last one is equal to zero from the other BDF6 conditions (see Table 5.1). So we get

$$
\begin{aligned}
&(1-6\beta)h_n^6 + \alpha_1 h_{n-1}^6 + \alpha_2(h_{n-1}+h_{n-2})^6 + \alpha_3(h_{n-1}+h_{n-2}+h_{n-3})^6 \\
&+ \alpha_4(h_{n-1}+h_{n-2}+h_{n-3}+h_{n-4})^6 + \alpha_5(h_{n-1}+h_{n-2}+h_{n-3}+h_{n-4}+h_{n-5})^6 = 0,
\end{aligned}
$$

which is the last BDF6 condition.

We now have all the BDF conditions, which happen to be linear in in α_i and β. So in theory, we are able to solve this system with a symbolic computer algebra system. While this approach works, in our experience, most symbolic solvers are not able to simplify the increasing complicated coefficients nicely. Hence, we opt to solve these linear systems by hand. We describe our approach in Section 5.1.2. A nice consequence of our approach is that we are able to retain the geometric intuition behind the formulas for the BDF coefficients (see Section 5.1.3). This intuition allows us to develop general formulas, applicable to all BDFs, that are novel to us. Having these formulas allows one to evaluate the BDF coefficients without the need to numerically

solve the linear system, which may be poorly conditioned.

5.1.2 Solving for the BDF Coefficients

BDF2

Although this system is rather easy to solve by hand, it is good to show our approach to solving the system of conditions in a simple setting. We begin by restating the BDF2 conditions

$$1 + \alpha_0 + \alpha_1 = 0, \tag{B1}$$

$$(1 - \beta)h_n - \alpha_1 h_{n-1} = 0, \tag{B2}$$

$$(1 - 2\beta)h_n^2 + \alpha_1 h_{n-1}^2 = 0. \tag{B3}$$

Multiplying (B2) by h_{n-1} and adding it to (B3) gives the system

$$1 + \alpha_0 + \alpha_1 = 0, \tag{B1}$$

$$(1 - \beta)h_n - \alpha_1 h_{n-1} = 0, \tag{B2}$$

$$\Big((1 - \beta)h_{n-1} + (1 - 2\beta)h_n\Big)h_n = 0, \tag{B3.1}$$

which can be rearranged to give

$$\beta = \frac{h_n + h_{n-1}}{h_{n-1} + 2h_n}.$$

We substitute β into (B2) to get

$$\alpha_1 = \frac{h_n^2}{h_{n-1}(h_{n-1} + 2h_n)}.$$

We then use (B1) to get

$$\alpha_0 = -\frac{(h_n + h_{n-1})^2}{h_{n-1}(h_{n-1} + 2h_n)}.$$

It is easy to check the we land at the same coefficients as in (4.14). This finishes our derivation for the BDF2 coefficients. Table 5.1 contains a list of all the BDF coefficients. We now proceed to solve for the BDF3 coefficients. We note that we will intentionally reuse the same numbering of the equation system above in the coming sections.

BDF3

The system of BDF3 conditions is

$$1 + \alpha_0 + \alpha_1 + \alpha_2 = 0, \tag{B1}$$

$$(1-\beta)h_n - \alpha_1 h_{n-1} - \alpha_2(h_{n-1} + h_{n-2}) = 0, \tag{B2}$$

$$(1-2\beta)h_n^2 + \alpha_1 h_{n-1}^2 + \alpha_2(h_{n-1} + h_{n-2})^2 = 0, \tag{B3}$$

$$(1-3\beta)h_n^3 - \alpha_1 h_{n-1}^3 - \alpha_2(h_{n-1} + h_{n-2})^3 = 0. \tag{B4}$$

Carrying the row reduction operations $(h_{n-1} + h_{n-2})\cdot$(B3) + (5.3) and $(h_{n-1} + h_{n-2})\cdot$(B2) + (5.2) give the system

$$1 + \alpha_0 + \alpha_1 + \alpha_2 = 0, \tag{B1}$$

$$(1-\beta)h_n - \alpha_1 h_{n-1} - \alpha_2(h_{n-1} + h_{n-2}) = 0, \tag{B2}$$

$$(1-\beta)h_n(h_{n-1} + h_{n-2}) + (1-2\beta)h_n^2 - \alpha_1 h_{n-1}h_{n-2} = 0, \tag{B3.1}$$

$$(1-2\beta)h_n^2(h_{n-1} + h_{n-2}) + (1-3\beta)h_n^3 + \alpha_1 h_{n-1}^2 h_{n-2} = 0. \tag{B4.1}$$

Furthermore, $h_{n-1}\cdot$ (B3.1) + (B4.1) yields the system

$$1 + \alpha_0 + \alpha_1 + \alpha_2 = 0, \quad \text{(B1)}$$

$$(1-\beta)h_n - \alpha_1 h_{n-1} - \alpha_2(h_{n-1} + h_{n-2}) = 0, \quad \text{(B2)}$$

$$(1-\beta)h_n(h_{n-1} + h_{n-2}) + (1-2\beta)h_n^2 - \alpha_1 h_{n-1}h_{n-2} = 0, \quad \text{(B3.1)}$$

$$\Big((1-\beta)h_{n-1}(h_{n-1} + h_{n-2}) + (1-2\beta)h_n(2h_{n-1} + h_{n-2}) + (1-3\beta)h_n^2\Big)h_n = 0. \quad \text{(B4.2)}$$

β can now be determined from (B4.2) to be

$$\beta = \frac{(h_n + h_{n-1})(h_n + h_{n-1} + h_{n-2})}{h_{n-1}(h_{n-1} + h_{n-2}) + 2h_n\Big(h_{n-1} + (h_{n-1} + h_{n-2})\Big) + 3h_n^2},$$

and, consequently, α_1 can be determined from (B3.1) to be

$$\alpha_1 = \frac{h_n^2(h_n + h_{n-1} + h_{n-2})^2}{h_{n-1}h_{n-2}\left(h_{n-1}(h_{n-1} + h_{n-2}) + 2h_n\Big(h_{n-1} + (h_{n-1} + h_{n-2})\Big) + 3h_n^2\right)}.$$

Similarly, α_2 and, in turn, α_0 will be determined from (B2) and (B1), respectively, to be

$$\alpha_2 = -\frac{h_n^2(h_n + h_{n-1})^2}{(h_{n-1} + h_{n-2})h_{n-2}\left(h_{n-1}(h_{n-1} + h_{n-2}) + 2h_n\Big(h_{n-1} + (h_{n-1} + h_{n-2})\Big) + 3h_n^2\right)},$$

$$\alpha_0 = -\frac{(h_n + h_{n-1})^2(h_n + h_{n-1} + h_{n-2})^2}{h_{n-1}(h_{n-1} + h_{n-2})\left(h_{n-1}(h_{n-1} + h_{n-2}) + 2h_n\Big(h_{n-1} + (h_{n-1} + h_{n-2})\Big) + 3h_n^2\right)},$$

which concludes our derivation of the BDF4 coefficients.

BDF4-6

Deriving the coefficients of the higher order BDFs gets quite cumbersome, however, it follows the exact same procedure. For the sake of readability, instead of fully deriving these coefficients as we did with BDF2 and BDF3, we only highlight how to do so for the rest of BDFs and leave the verification for the interested reader. We do, however, include all BDF coefficients in Table 5.1.

Figure 5.1 illustrates how the proposed technique generalizes to higher order BDFs. In particular, it includes a diagram showing which equations are involved in the row reductions, when variables are eliminated, and which system of equations will eventually be used to determine the coefficients. For example, a scaled version of B4 and B5 will be used to obtain B5.1, which does not involve the variable α_3. In turn, using B4.1 and B4.2 allows us to eliminate α_2 and α_1, respectively, and obtain B5.3, which only involves β. The system of equations that will eventually be used to determine the coefficients will consist of those equations appearing at the end of each row. In the case of BDF4, that is B1,B2,B3.1,B4.2, and B5.3.

We also see that the proposed elimination procedure is nested as shown in the compact diagram in Fig. 5.1. By nested, we mean that after the first set of row reductions for, say BDF6, one can simply follow the same row reduction steps, albeit on a subset of the resulting system of equations and with different weights, as BDF5 to eventually solve the system of equations. This was shown in our derivation of the BDF3 coefficients. After the first set of row-reductions, we obtained the system (B1), (B2), (B3.1), and (B4.1). The procedure we applied on the subsystem consisting of (B2), (B3.1), and (B4.1) to obtain β, α_1, and α_2 is the same as the one we followed to solve for the BDF2 coefficients.

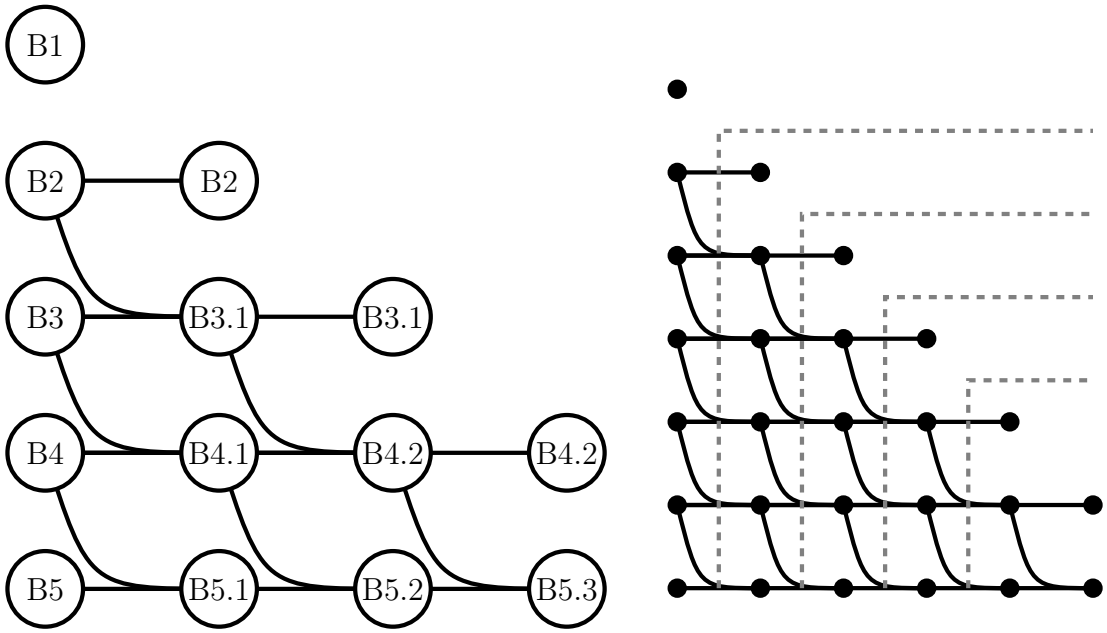


Figure 5.1: Left: The proposed row reduction scheme for solving for the BDF4 coefficients with the same naming convention. Each set of row reductions results in the elimination of a variable from the new equations. The last equation from each row will be used to determine the coefficients. Right: A compact version of the proposed row reduction scheme for solving for all the BDF coefficients. The scheme is nested in the sense that after the first set of row reductions for BDFk, one can complete solving the system using the same row reduction steps, albeit with different coefficients, as BDF$(k-1)$. The gray dashed lines indicate the boundaries of the block of a lower order BDF. The nested blocks in order of increasing sizes are for the BDFs of increasing order, respectively.

Table 5.1: BDFk conditions and coefficients for $k = 2, \ldots, 6$

Method	Conditions	Coefficients	
BDF2	$1 + \alpha_0 + \alpha_1 = 0$	β	$\frac{h_n + h_{n-1}}{h_{n-1} + 2h_n}$
	$(1-\beta)h_n - \alpha_1 h_{n-1} = 0$	α_0	$-\frac{(h_n + h_{n-1})^2}{h_{n-1}(h_{n-1} + 2h_n)}$
	$(1-2\beta)h_n^2 + \alpha_1 h_{n-1}^2$	α_1	$\frac{h_n^2}{h_{n-1}(h_{n-1} + 2h_n)}$
BDF3	$1 + \alpha_0 + \alpha_1 + \alpha_2 = 0$	β	$(h_n + h_{n-1})(h_n + h_{n-1} + h_{n-2})\Big(h_{n-1}(h_{n-1} + h_{n-2}) + 2h_n\Big(h_{n-1} + (h_{n-1} + h_{n-2})\Big) + 3h_n^2\Big)$
	$(1-\beta)h_n - \alpha_1 h_{n-1} - \alpha_2(h_{n-1} + h_{n-2}) = 0$	α_0	$-\frac{(h_n + h_{n-1})^2(h_n + h_{n-1} + h_{n-2})^2}{h_{n-1}(h_{n-1} + h_{n-2})}\Big(h_{n-1}(h_{n-1} + h_{n-2}) + 2h_n\Big(h_{n-1} + (h_{n-1} + h_{n-2})\Big) + 3h_n^2\Big)^{-1}$
	$(1-2\beta)h_n^2 + \alpha_1 h_{n-1}^2 + \alpha_2(h_{n-1} + h_{n-2})^2 = 0$	α_1	$\frac{h_n^2(h_n + h_{n-1} + h_{n-2})^2}{h_{n-1}h_{n-2}}\Big(h_{n-1}(h_{n-1} + h_{n-2}) + 2h_n\Big(h_{n-1} + (h_{n-1} + h_{n-2})\Big) + 3h_n^2\Big)^{-1}$
	$(1-3\beta)h_n^3 - \alpha_1 h_{n-1}^3 - \alpha_2(h_{n-1} + h_{n-2})^3 = 0$	α_2	$-\frac{h_n^2(h_n + h_{n-1})^2}{(h_{n-1} + h_{n-2})h_{n-2}}\Big(h_{n-1}(h_{n-1} + h_{n-2}) + 2h_n\Big(h_{n-1} + (h_{n-1} + h_{n-2})\Big) + 3h_n^2\Big)^{-1}$
BDF4	See Table 5.2		
BDF5			
BDF6			

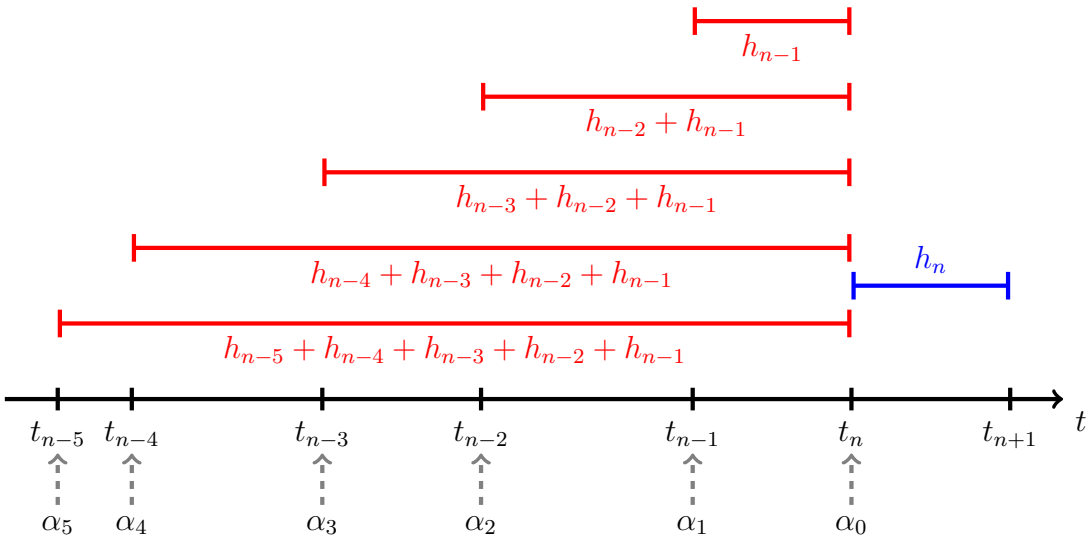


Figure 5.2: Geometry of the BDF setup. The various backward steps are shown red, while the forward step is shown in blue. It will be useful to associate each α_i to the corresponding time node it appears with (see (5.1) to (5.5)) as shown here.

5.1.3 A General Formula for the BDF Coefficients

Going through the battle of manually solving the system of BDF conditions allowed us to obtain simplified expressions for the BDF coefficients (see Table 5.1). This triumph enables us to notice a connection between the formulas and the geometry of the BDFs, as is indicated in the colour-coded BDF formulas in Table 5.1. Using algebraic-geometric connection we are able to collect our prize, general formulas for all the BDF coefficients for all orders.

Before we begin our discussion, it is useful to categorize steps based on the direction from the point t_n, as shown in Fig. 5.2. For example, in the case of BDF2 we have one backward step of size h_{n-1} and one forward step of size h_n. It is also useful to associate the α_i's with their corresponding node as we illustrate in Fig. 5.2. For example, as α_i appears as the weight of x_{n-i} in (5.1) to (5.5), we will associate it with time t_{n-i}.

We begin our generalization with β (see Table 5.1). As it is written, it can be seen that the denominator of β contains terms with increasing powers of the forward step h_n with product of some combination of backward steps, so that the terms have

$O(h^{k-1})$. Thus, the denominator of the β coefficients for all BDFs is

$$\sum_{j=0}^{k-1}(j+1)h_n^j\left(\sum_{\substack{H\subset\mathcal{B}\\|H|=k-1-j}}\prod_{h\in H}h\right),$$

in which $\mathcal{B}$ is defined to be the set of all backward steps. We can also see that the numerator happens to be the product of all the backward steps concatenated with the forward step, or in other words

$$\prod_{j=1}^{k-1}(t_{n+1}-t_{n-j}).$$

Hence, we get

$$\beta=\frac{\prod_{j=1}^{k-1}(t_{n+1}-t_{n-j})}{\sum_{j=0}^{k-1}(j+1)h_n^j\left(\sum_{\substack{H\subset\mathcal{B}\\|H|=k-1-j}}\prod_{h\in H}h\right)},$$

for all the BDFs.

As for the α_i's, their denominators are composed of two terms, one of which is the denominator of β. To see the geometric connection with the other part, we reiterate the usefulness of associating each α_i to a time node as shown in Fig. 5.2. From that lens, it can be seen that the other term in the denominator is the distance from the time node associated to α_i to all other historical time nodes, or in other words

$$\prod_{\substack{j=0\\j\neq i}}^{k-1}(t_{n-i}-t_{n-j}).$$

By historical time nodes, we mean $t_{n-(k-1)},\ldots,t_n$, which correspond to the known

values x_{n-i}, as opposed to the one we are determining, x_{n+1}, at t_{n+1}. As for the numerator of the α_i's, it is quite similar to that of β, with the exception of the power and of skipping the factor which involves the time node t_{n-i}. In other words, the numerator is given by

$$\prod_{\substack{j=1\\ j\neq i}}^{k-1}(t_{n+1}-t_{n-j})^2.$$

All in all we get

$$\alpha_i = \frac{\displaystyle\prod_{\substack{j=1\\ j\neq i}}^{k-1}(t_{n+1}-t_{n-j})^2}{\left(\displaystyle\prod_{\substack{j=0\\ j\neq i}}^{k-1}(t_{n-i}-t_{n-j})\right)\left(\displaystyle\sum_{j=0}^{k-1}(j+1)h_n^j\left(\sum_{\substack{H\subset\mathcal{B}\\ |H|=k-1-j}}\prod_{h\in H}h\right)\right)},$$

which can also be stated as

$$\alpha_i = (-1)^{i+1}\frac{\displaystyle\prod_{\substack{j=1\\ j\neq i}}^{k-1}(t_{n+1}-t_{n-j})^2}{\left(\displaystyle\prod_{\substack{j=0\\ j\neq i}}^{k-1}|t_{n-i}-t_{n-j}|\right)\left(\displaystyle\sum_{j=0}^{k-1}(j+1)h_n^j\left(\sum_{\substack{H\subset\mathcal{B}\\ |H|=k-1-j}}\prod_{h\in H}h\right)\right)},$$

for $i=0,\ldots,k-1$ in the case of BDFk.

Thus, we are able to distill all of Table 5.1 into a much more compact Table 5.2. It is possible to simplify this further by having one formula for all α_i's and β but it overcomplicates the resulting formula. We also find it to be unnecessary and, given the context of the coefficients, unnatural.

Table 5.2: A compact summary of the general BDFk conditions and coefficients that apply for all $k = 2, \ldots, 6$

Method	Conditions	Coefficients	
BDFk	$1 + \sum_{i=0}^{k-1} \alpha_i = 0$	β	$\dfrac{\prod_{j=1}^{k-1}(t_{n+1} - t_{n-j})}{\sum_{j=0}^{k-1}(j+1)h_n^j \left(\sum_{\substack{H \subset \mathcal{B} \\ \lvert H \rvert = k-1-j}} \prod_{h \in H} h \right)}$
	$(1-j\beta)h_n^j + (-1)^j \sum_{i=1}^{k-1} \alpha_i \left(\sum_{\ell=1}^{i} h_{n-\ell} \right)^j = 0$, for $j = 1, \cdots, k$	α_i	$(-1)^{i+1} \dfrac{\prod_{\substack{j=0 \\ j \neq i}}^{k-1}(t_{n+1} - t_{n-j})^2}{\left(\prod_{\substack{j=0 \\ j \neq i}}^{k-1} \lvert t_{n-i} - t_{n-j} \rvert \right) \left(\sum_{j=0}^{k-1}(j+1)h_n^j \left(\sum_{\substack{H \subset \mathcal{B} \\ \lvert H \rvert = k-1-j}} \prod_{h \in H} h \right) \right)}$, for $i = 1, \ldots, k-1$

5.2 Higher Order Adaptive Time-Stepper

We are now ready to begin generalizing the adaptive timer-stepper to the higher order BDFs. We will begin with describing the generalized time-stepping scheme and move on to deriving the appropriate error bounds as was done in Section 4.2.

5.2.1 The General Time-Stepping Scheme

We choose our general time-stepping scheme to be a natural extension of the time-stepping scheme described in Section 4.2.1. More concretely, our general time-stepping scheme will involve a coarse and a fine approximation that allow us to estimate the coarse local truncation error. The only difference of the general method to the one previously described is using more historical points for the higher order BDFs. In particular, we also retain the memory efficiency of the scheme, which was described in Section 4.2.7. Since the general time-stepping scheme is such a natural extension of the original one, we are able to depict it for all the BDFs at once (see Fig. 5.3).

5.2.2 Local Truncation Error Estimate for the General Time-Stepper

In this section we will find and prove an LTE estimate for the general time-stepper. We start by addressing a few notational issues that will arise in the course of the proof. First, from Table 5.1, it is clear that the α_i's and β are functions of the step-sizes taken. Hence, we can refer to them as $\alpha_i(h_n, h_{n-1}, \ldots, h_{n-(k-1)})$ and $\beta(h_n, h_{n-1}, \ldots, h_{n-(k-1)})$ when necessary to indicate the step-sizes. We let $\alpha_i = \alpha_i(h_n, h_{n-1}, \ldots, h_{n-(k-1)})$, $\alpha_i' = \alpha_i(\frac{h_n}{2}, h_{n-1}, \ldots, h_{n-(k-1)})$, and $\alpha_i'' = \alpha_i(\frac{h_n}{2}, \frac{h_n}{2}, h_{n-1}, \ldots, h_{n-(k-2)})$ and similarly define β, β', and β''. In particular, the (α_i, β), (α_i', β'), and (α_i'', β'') are the BDF coefficients associated to the coarse step, first fine, and second fine step, respectively (see Fig. 5.3). Furthermore, we recall that x_n will refer to the approximation of $x(t_n)$ obtained using

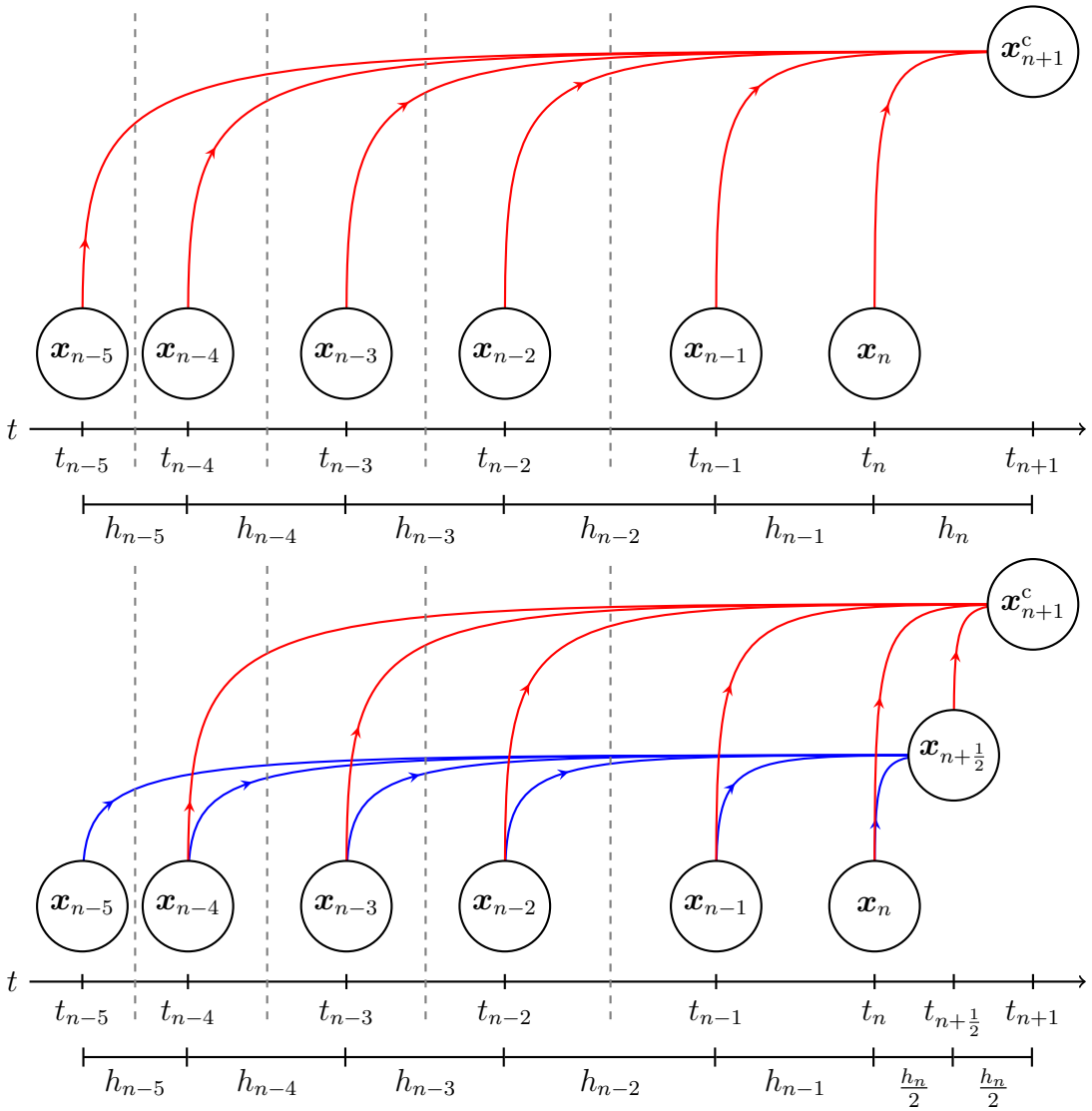


Figure 5.3: Schematic depiction for the general time-stepper for BDFk, $k = 2\ldots,6$. Top: The coarse approximation of $\boldsymbol{x}$ at t_{n+1}, $\boldsymbol{x}^{\mathrm{c}}_{n+1}$, is computed using k accepted values, $\boldsymbol{x}_{n-(k-1)},\ldots,\boldsymbol{x}_n$, which were precomputed at previous time-steps. Bottom: The fine approximation of $\boldsymbol{x}$ at t_{n+1}, $\boldsymbol{x}^{\mathrm{f}}_{n+1}$, is computed in two steps. First we compute the intermediate value $\boldsymbol{x}_{n+\frac{1}{2}}$, which approximates $\boldsymbol{x}$ at $t_{n+\frac{1}{2}} = t_n + \frac{h_n}{2}$ (shown in blue). Then we compute $\boldsymbol{x}^{\mathrm{f}}_{n+1}$ using $\boldsymbol{x}_{n-(k-2)},\ldots,\boldsymbol{x}_n$ and $\boldsymbol{x}_{n+\frac{1}{2}}$ (shown in red). The dashed gray lines are to indicate the scheme for the various BDFs. For example, this diagram depicts the time-stepping scheme for BDF6 if we consider all time values, while it does so for BDF5 if we ignore the time values to the left of the dashed line (i.e. ignore t_{n-5}).

one of the BDFs. For the rest of the proof, since we are interested in local estimates, we assume $x_n, \ldots, x_{n-(k-1)}$ to be known and thus are equal to $x(t_n), \ldots, x(t_{n-(k-1)})$, respectively. We only make the distinction between the approximate and exact values for x_{n+1} and $x_{n+\frac{1}{2}}$. Also, we recall that $\frac{dx}{dt} = f(t, x)$ and note that we will be assuming the existence of as many derivatives as needed for the rest of this section.

The following lemma is essential to computing the LTE for the various BDFs and can be found in good numerical analysis textbooks, for example [53].

Lemma 5.1. *For a fixed value $t \in \mathbb{R}$, $f(t, x_1)$ and $f(t, x_2)$ are equal up to the same order as x_1 and x_2.*

Proof. In the case of scalar functions the above lemma follows from an application of the mean value theorem on the function $g(x) = f(t, x)$, for the fixed value t. For vector-valued functions we simply repeat the above component-wise. □

Lemma 5.2. *The LTE for the variable BDFk is of order $k + 1$. Furthermore, the lowest order error term of the LTE is*

$$\frac{1}{(k+1)!}\left((1-(k+1)\beta)h_n^{k+1} + (-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i\left(\sum_{\ell=1}^{i}h_{n-\ell}\right)^{k+1}\right)x^{(k+1)}(t_n).$$

Proof. By definition of the LTE and (5.1) to (5.5)

$$\begin{aligned}\text{LTE} &= x(t_{n+1}) - x_{n+1} \\ &= x(t_{n+1}) - \left(-\left(\sum_{i=0}^{k-1}\alpha_i x_{n-i}\right) + h_n\beta f(t_{n+1}, x_{n+1})\right). \end{aligned} \tag{5.13}$$

By Lemma 5.1, we can simply replace $f(t_{n+1}, x_{n+1})$ by $f(t_{n+1}, x(t_{n+1}))$ as the h_n factor means it will not affect the lowest order error term. Furthermore, using a

Taylor series about t_n, we obtain that

$$1 + \sum_{i=0}^{k-1} \alpha_i$$

is the zeroth order term and that

$$\frac{1}{j!}\left((1-j\beta)h_n^j + (-1)^j \sum_{i=1}^{k-1} \alpha_i \left(\sum_{\ell=1}^{i} h_{n-\ell}\right)^j\right) x^j(t_n),$$

for $j = 1, \cdots, k+1$ are the order $1, \ldots, k+1$ terms, respectively. The desired results follow from noting that all these terms, with the exception of the $k+1$ order term, are forced to be zero by our assumptions for BDFk (see Table 5.2). □

Using Lemma 5.2, we can derive the coarse LTE, ϵ_{c}, and fine LTE, ϵ_{f}.

Theorem 5.3. *For BDFk, the coarse and fine LTEs are given by*

$$\begin{aligned}
\epsilon_{\mathrm{c}} &= \frac{1}{(k+1)!}\left((1-(k+1)\beta)h_n^{k+1} + (-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i\left(\sum_{\ell=1}^{i} h_{n-\ell}\right)^{k+1}\right)x^{(k+1)}(t_n) \\
&\quad + O(h^{k+2}), \qquad (5.14) \\
\epsilon_{\mathrm{f}} &= \frac{1}{(k+1)!}\left[\left((1-(k+1)\beta'')\left(\frac{h_n}{2}\right)^{k+1} + (-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i''\left(\sum_{\ell=1}^{i} h_{n-\ell}''\right)^{k+1}\right)\right. \\
&\quad \left. -\alpha_0''\left((1-(k+1)\beta')\left(\frac{h_n}{2}\right)^{k+1} + (-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i'\left(\sum_{\ell=1}^{i} h_{n-\ell}\right)^{k+1}\right)\right]x^{(k+1)}(t_n) \\
&\quad + O(h^{k+2}), \qquad (5.15)
\end{aligned}$$

in which $h_{n-1}'' = \frac{h_n}{2}$ *and* $h_{n-\ell}'' = h_{n-(\ell-1)}$ *for* $\ell = 2, \ldots, k-1$.

Proof. In light of Lemma 5.2, it suffices to only prove (5.15). Using Lemma 5.2 and (5.13) we see that

$$
\begin{aligned}
&- \left(\sum_{i=1}^{k-1} \alpha_i x_{n-i}\right) + h_n \beta f(t_{n+1}, x_{n+1}) = x(t_{n+1}) + \alpha_0 x_n \\
&\quad - \frac{1}{(k+1)!}\left((1-(k+1)\beta)h_n^{k+1} + (-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i\left(\sum_{\ell=1}^{i} h_{n-\ell}\right)^{k+1}\right)x^{(k+1)}(t_n) \\
&\qquad\qquad + O(h^{k+2}).
\end{aligned}
$$

Similarly, for the first fine step we get

$$
\begin{aligned}
&- \left(\sum_{i=1}^{k-1} \alpha_i' x_{n-i}\right) + \frac{h_n}{2} \beta' f(t_{n+\frac{1}{2}}, x_{n+\frac{1}{2}}) = x(t_{n+\frac{1}{2}}) + \alpha_0' x_n \\
&- \frac{1}{(k+1)!}\left((1-(k+1)\beta')\left(\frac{h_n}{2}\right)^{k+1} + (-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i'\left(\sum_{\ell=1}^{i} h_{n-\ell}\right)^{k+1}\right)x^{(k+1)}(t_n) \\
&\qquad\qquad + O(h^{k+2}). \quad (5.16)
\end{aligned}
$$

Assuming we apply BDFk to obtain x_{n+1} from $x(t_{n+\frac{1}{2}}), x_n, \ldots, x_{n-(k-2)}$, we similarly get

$$
\begin{aligned}
&- \left(\sum_{i=1}^{k-1} \alpha_i'' x_{n-(i-1)}\right) + \frac{h_n}{2} \beta'' f(t_{n+1}, x_{n+1}) = x(t_{n+1}) + \alpha_0'' x(t_{n+\frac{1}{2}}) \\
&- \frac{1}{(k+1)!}\left((1-(k+1)\beta'')\left(\frac{h_n}{2}\right)^{k+1} + (-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i''\left(\sum_{\ell=1}^{i} h''_{n-\ell}\right)^{k+1}\right)x^{(k+1)}(t_n) \\
&\qquad\qquad + O(h^{k+2}). \quad (5.17)
\end{aligned}
$$

Now by definition of ϵ_{f}, we have

$$
\begin{aligned}
\epsilon_{\mathrm{f}} &= x(t_{n+1}) - x_{n+1}^{\mathrm{f}} \\
&= x(t_{n+1}) - \left[-\alpha_0'' x_{n+\frac{1}{2}} - \left(\sum_{i=1}^{k-1} \alpha_i'' x_{n-(i-1)}\right) + \frac{h_n}{2}\beta'' f(t_{n+1}, x_{n+1})\right] \\
&= x(t_{n+1}) - \left[-\alpha_0''\left(-\alpha_0' x_n - \left(\sum_{i=1}^{k-1} \alpha_i' x_{n-i}\right) + \frac{h_n}{2}\beta' f(t_{n+\frac{1}{2}}, x_{n+\frac{1}{2}})\right)\right.
\end{aligned}
$$

$$-\left(\sum_{i=1}^{k-1}\alpha_i'' x_{n-(i-1)}\right)+\frac{h_n}{2}\beta'' f(t_{n+1},x_{n+1})\Bigg]. \tag{5.18}$$

Substituting (5.16) and (5.17) into (5.18) completes the proof. □

We are now in a position to state and prove our LTE error estimate based on the approximations x_{n+1}^{c} and x_{n+1}^{f}.

Theorem 5.4. *The coarse LTE can be stated without the use of derivatives as*

$$\begin{aligned}\epsilon_{\mathrm{c}} = &\left[(1-(k+1)\beta)h_n^{k+1}+(-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i\left(\sum_{\ell=1}^{i}h_{n-\ell}\right)^{k+1}\right]\\
&\Bigg[\left((1-(k+1)\beta'')\left(\frac{h_n}{2}\right)^{k+1}+(-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i''\left(\sum_{\ell=1}^{i}h_{n-\ell}''\right)^{k+1}\right)\\
&-\alpha_0''\left((1-(k+1)\beta')\left(\frac{h_n}{2}\right)^{k+1}+(-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i'\left(\sum_{\ell=1}^{i}h_{n-\ell}\right)^{k+1}\right)\\
&-\left((1-(k+1)\beta)h_n^{k+1}+(-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i\left(\sum_{\ell=1}^{i}h_{n-\ell}\right)^{k+1}\right)\Bigg]^{-1}\left(x_{n+1}^{\mathrm{c}}-x_{n+1}^{\mathrm{f}}\right)\\
&+O(h^{k+2}).\end{aligned} \tag{5.19}$$

Proof. We know that $x_{n+1}^{\mathrm{c}}-x_{n+1}^{\mathrm{f}}=\epsilon_{\mathrm{f}}-\epsilon_{\mathrm{c}}$. Substituting in (5.14) and (5.15) and solving for $x^{(k+1)}(t_n)$ gives

$$\begin{aligned}x^{(k+1)}(t_n) = &\left[(k+1)!\left(x_{n+1}^{\mathrm{c}}-x_{n+1}^{\mathrm{f}}\right)\right]\\
&\Bigg[\left((1-(k+1)\beta'')\left(\frac{h_n}{2}\right)^{k+1}+(-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i''\left(\sum_{\ell=1}^{i}h_{n-\ell}''\right)^{k+1}\right)\\
&-\alpha_0''\left((1-(k+1)\beta')\left(\frac{h_n}{2}\right)^{k+1}+(-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i'\left(\sum_{\ell=1}^{i}h_{n-\ell}\right)^{k+1}\right)\\
&-\left((1-(k+1)\beta)h_n^{k+1}+(-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i\left(\sum_{\ell=1}^{i}h_{n-\ell}\right)^{k+1}\right)\Bigg]^{-1}+O(h),\end{aligned}$$

which gives the desired results once plugged back into (5.14). □

Corollary 5.5. *Richardson Extrapolation of the generalized adaptive time-stepper takes the following linear combination of* x^{c}_{n+1} *and* x^{f}_{n+1}

$$\begin{aligned} x(t_{n+1}) = &\left[\left((1-(k+1)\beta'')\left(\frac{h_n}{2}\right)^{k+1} + (-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i''\left(\sum_{\ell=1}^{i}h''_{n-\ell}\right)^{k+1}\right)\right. \\ &\left.-\alpha_0''\left((1-(k+1)\beta')\left(\frac{h_n}{2}\right)^{k+1} + (-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i'\left(\sum_{\ell=1}^{i}h_{n-\ell}\right)^{k+1}\right)\right] D^{-1}x^{\mathrm{c}}_{n+1} \\ &-\left[(1-(k+1)\beta)h_n^{k+1} + (-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i\left(\sum_{\ell=1}^{i}h_{n-\ell}\right)^{k+1}\right] D^{-1}x^{\mathrm{f}}_{n+1}, \end{aligned}$$

in which

$$\begin{aligned} D = &\left((1-(k+1)\beta'')\left(\frac{h_n}{2}\right)^{k+1} + (-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i''\left(\sum_{\ell=1}^{i}h''_{n-\ell}\right)^{k+1}\right) \\ &-\alpha_0''\left((1-(k+1)\beta')\left(\frac{h_n}{2}\right)^{k+1} + (-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i'\left(\sum_{\ell=1}^{i}h_{n-\ell}\right)^{k+1}\right) \\ &-\left((1-(k+1)\beta)h_n^{k+1} + (-1)^{k+1}\sum_{i=1}^{k-1}\alpha_i\left(\sum_{\ell=1}^{i}h_{n-\ell}\right)^{k+1}\right). \end{aligned}$$

Proof. This follows directly from setting $x(t_{n+1}) = x^{\mathrm{c}}_{n+1} + \epsilon_{\mathrm{c}}$ (see Section 4.2.3). □

5.3 Comparison of the Adaptive Time-Stepper of Various Orders

We use the initial value problem (4.18) to (4.20), on the same time interval $[0, 30]$, to compare our adaptive time-steppers of various orders. Figure 5.4 shows the global error for our adaptive time-steppers for orders $k = 2, \ldots, 6$. In addition to achieving smaller global error values, the higher order time-steppers do so with much larger

Table 5.3: Number of steps taken to solve the initial value problem (4.18) to (4.20) by the adaptive time-steppers based on the various BDFs.

Method	Number of steps
BDF2	5708
BDF3	1793
BDF4	942
BDF5	639
BDF6	525

time-steps. Indeed the safety mechanism (see Section 4.2.6) prevented the time-steppers based on BDF5 and BDF6 from taking bigger step-sizes. This occurred more frequently for the time-stepper based on BDF6. By taking larger step-sizes, higher order time-steppers took significantly fewer time-steps as shown in Table 5.3. BDF6 needed a tenth of the number of steps that BDF2 needed to solve the initial value problem to the same tolerance.

However, we must mention that the stability of the higher order BDFs for variable step-sizes is a concern [53]. It appears to be dependent on the ratio of the step-sizes taken [53, 52]. For example Grigorieff [52] proved that variables step-size BDF2 is stable so long as the ratio of the step-sizes is in the interval $[0, 2.414]$ [53]. In comparison, he showed the intervals for BDF3-5 to be $[0.836, 1.127]$, $[0.979, 1.019]$, $[0.997, 1.003]$, respectively, with the bound for BDF6 to be a proper subset of $[0.999, 1.001]$ [52]! That said, this was proven without any assumptions on how the step-sizes vary [53, 52]. Some considerations have been given to the stability of the variable BDFs of higher order when extra assumptions, such as ensuring the rate of change of the step-size, are made [53, 47, 48].

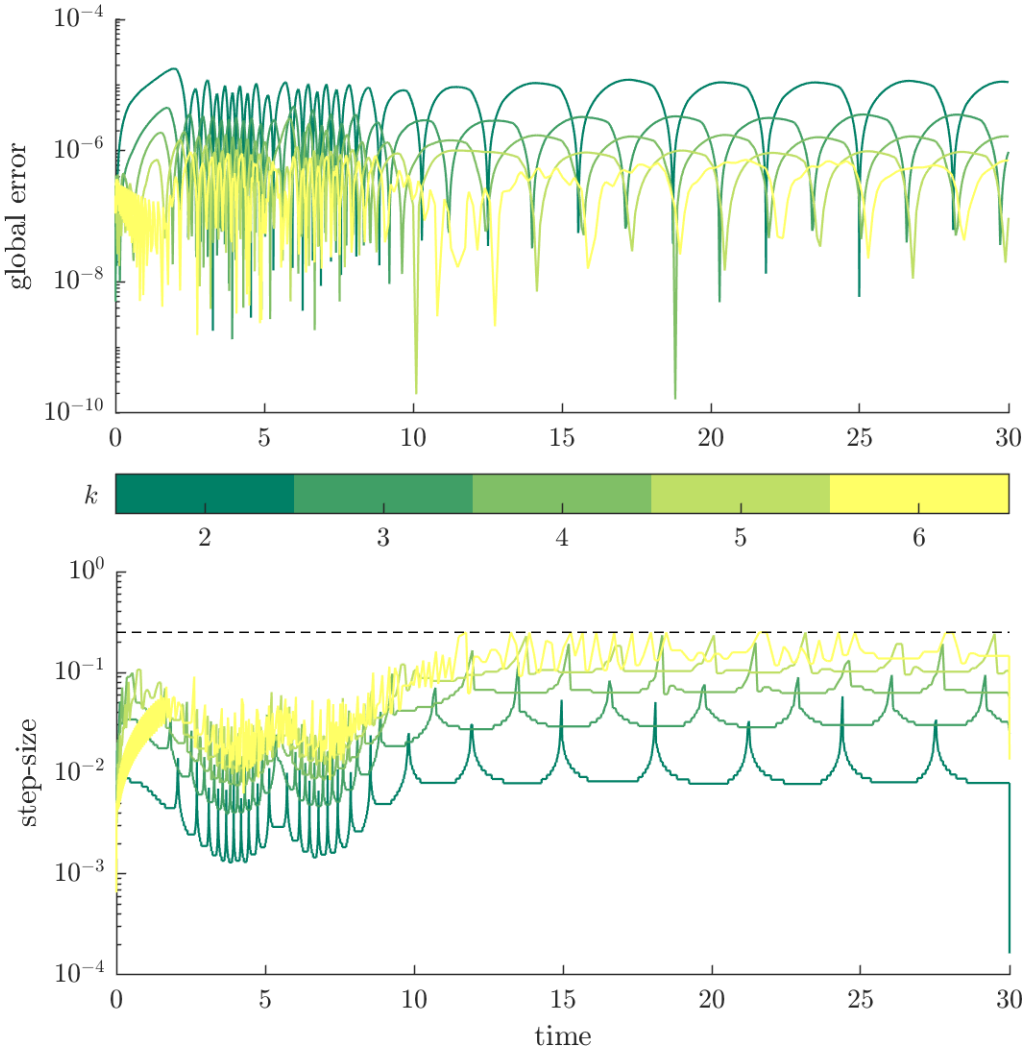


Figure 5.4: Top: Global error of the various adaptive time-steppers as a function of time for $tol = 10^{-7}$. Higher order adaptive time-steppers achieve lower global error than the lower order ones. The different colours indicate the order of the adaptive time-stepper as shown in the colour bar. Bottom: Step-size taken by the various adaptive time-steppers as a function of time. The dashed horizontal line is the maximum allowed step-size $\Delta t_{\max}$. Higher order time-steppers can take much larger step-sizes as compared to lower order ones and attain lower global error. The decision of an adaptive time stepper to take step-sizes larger than $\Delta t_{\max}$ is overridden by a safety feature. This is best observed for the BDF6 timer-stepper in this figure.

Chapter 6

Simulation Results

In this chapter we present and discuss the results of a variety of simulations with various experimental set-ups. Some of these findings were consistent with the literature while others have no experimental counterpart, but ought to have experiments designed to study them. For complete details of the various simulations see Appendix A.

6.1 Desensitization and a-Waves

In numerical simulation 3, the center of the retina is stimulated with spatially Gaussian ($\sigma = 50$), 20 ms light pulses at $t = 0, 2.0, 2.1, 2.2, \ldots, 2.9$ s. Our aim is to see what general features of the electroretinogram (ERG) and retinal physiology can be detected with our model. A 2 s gap provides time for a dark-adapted photoreceptor to return to resting potential. Figure 6.1 shows the induced change in potential and voltage throughout the surface of the eye and the retina, respectively. It also includes a plot of the potential as a function of time at a specified location in the eye. In the time plots of the potential, there are two very different time scales, a fast one occurring right after the light stimulus accompanied by hyperpolarization, and a very slow one in which the cell membrane returns to its depolarized equilibrium. This property is exactly what we are hoping to exploit in our adaptive time-stepper, since

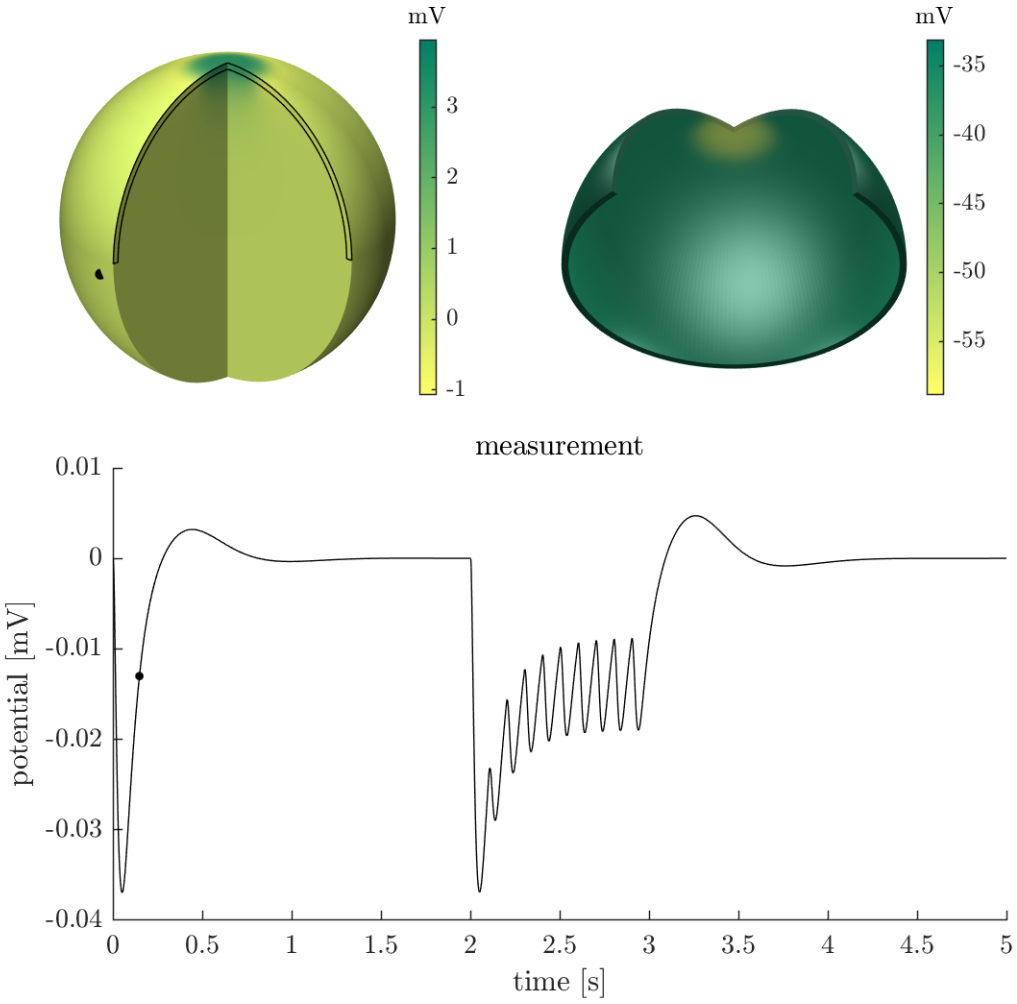


Figure 6.1: Top-left: Extracellular potential throughout the eye at $t = 0.15$ s. The region outlined in black is the retina. The black dot is the location of the potential measurement shown in the bottom plot. The effect of the light stimulus is observed in the darker green region near and around the center of the retina. Top-right: Voltage on the retina at $t = 0.15$ s. The hyperpolarization, as a result of the light stimulus, is observed in yellow in the central region of the retina. Bottom: Extracellular potential at the specified point (black dot in the top-left extracellular potential plot) on the surface of the eye as a function of time. There are two distinct phases observed in the resulting change from the stimulus, a fast hyperpolarization phase followed by a much slower depolarization phase. See numerical simulation 3 in Appendix A for more details about the set-up of this simulation.

in periods of slow change the solver may take much larger steps.

Many features of retinal physiology are present in our model, some of which are immediately clear from Fig. 6.1. For example, the voltage response between the flashes at time $t = 0$ s and $t = 3$ s is similar, while the response is weaker for the subsequent flashes. This is expected as there was an extended period of time between the first two flashes, while the remaining flashes were temporally close. This implies that the photoreceptors were able to recover from the first flash but later was desensitized to the rapid light stimuli and so the response was less pronounced.

The a-wave is another feature of retinal physiology that was present in our model. The two main components of a human ERG reading are the a-wave and the b-wave [94]. The a-wave is hypothesized to occur as a result of the hyperpolarization of the photoreceptors [94], and hence we were able to observe it in the surface of eye measurements shown in Fig. 6.1. However, the b-wave is hypothesized to occur as a result of the depolarization of retinal cells that are postsynaptic to photoreceptors [94], which are not included in the present model, so we were not able to detect them using our model. This is a confirmation of the proposed mechanism for the a-waves and b-waves. The overshoot observed as the membrane returns to resting potential in Fig. 6.1 is different, and much smaller in amplitude, than the typical b-wave [94].

6.2 Calcium in Photoreceptors

The model can be used to study other aspects of the retina as well, depending on the level of detail of the chosen transmembrane current model. For instance, the Kamiyama model we chose distinguishes between the calcium concentrations at different locations in the photoreceptor. It defines distinct auxiliary variables for the concentrations of the outer segment calcium, [Ca], inner segment submembrane cal-

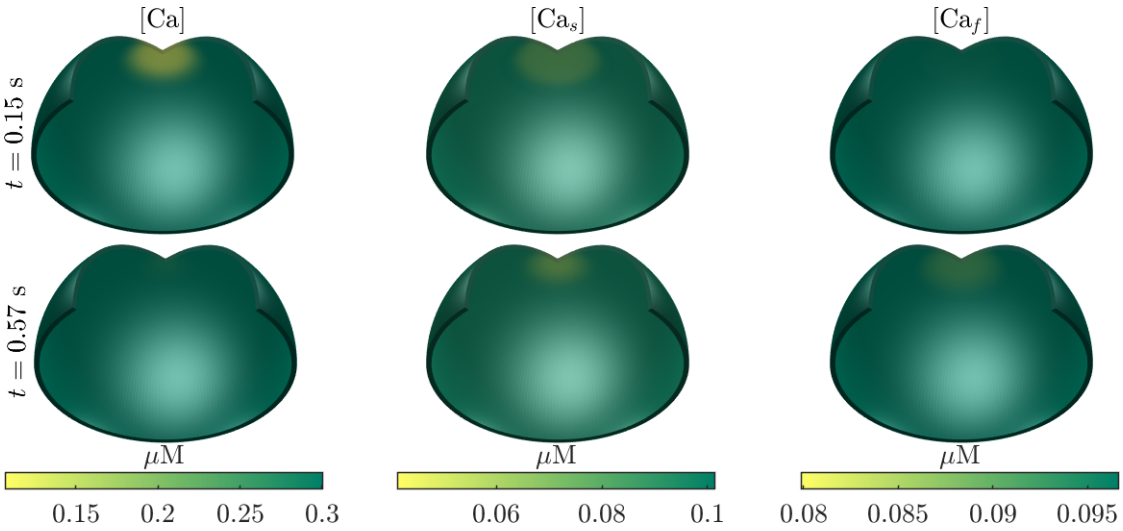


Figure 6.2: Top: The concentration of the outer segment calcium, [Ca], (left), inner segment submembrane calcium, [Ca$_s$], (center), inner segment central space calcium, [Ca$_f$], (right) at $t = 0.15$ s. The associated colour bar of each plot is located at the bottom of the plot's column. The outer segment calcium is quicker to respond to light stimuli than the inner segment calcium. As the movement of calcium is buffered, central space calcium response is not yet visible. Bottom: similar to the top row but at $t = 0.57$ s. Inner segment calcium response to light stimuli is longer lasting than that of the outer segment. There are no observed differences between submembrane and central space calcium in terms of duration of response.

cium, [Ca$_s$], and inner segment central space calcium, [Ca$_f$]. So, using the aforementioned numerical simulation 3, we are able to study calcium in both the outer segment and the inner segment of the photoreceptors. Figure 6.2 shows [Ca], [Ca$_s$], and [Ca$_f$] at times $t = 0.15$ s and $t = 0.57$ s. The distinction between the submembrane calcium and central space calcium reflects research showing that the movement of calcium is controlled in cells using various buffers [82]. Using the model, we observe the time delay between the calcium response to the light stimulus in the outer and inner segment. This is expected as the outer segment is the site of photon absorption, which marks the beginning of phototransduction [35]. Furthermore, we observe the delay in the response between the submembrane and central space calcium due to the buffers action on calcium. We also observe that the response in the outer segment is longer lasting than that of the inner segment, which has yet to be observed experimentally.

6.3 Gap Junctional Effect on Inner Segment Calcium Concentration

In numerical simulation 4, we use the multi-domain framework to model four active domains (rods and long (L-), medium (M-), and short wavelength cones (S-cones)). Each active domain is given a disjoint light stimulus at distinct times (see Appendix A). In reality, it may not be possible to give completely disjoint light stimuli to all the different kinds of cones, since, for example, the L-cones and M-cones have significant overlap in the frequencies of light they are sensitive to [72]. However, one can essentially do so for rods and cones by using light stimuli of different frequencies [26]. The goal of this simulation study is to see whether the various calcium concentrations in an active domain can be affected solely through gap junctions.

Our findings indicate that this is possible for the inner segment calcium concentrations only. Figure 6.3 shows the various calcium concentrations of the L-cones domain. It shows that even in the absence of a light stimuli to the L-cones domain, both the inner segment submembrane calcium concentration and the inner segment central space calcium concentration are affected by the light stimuli to the rods domain. However, no change was detected in the outer segment calcium concentration. While we found no experimental evidence supporting these observations, we believe they will hold considering that calcium plays a key role in phototransduction (contributing about a fourth of the photocurrent) [118]. The observations are also consistent with the findings that gap junctions occur far away from the outer segments, with some occurring in the inner segments [95, 70, 115, 24, 72].

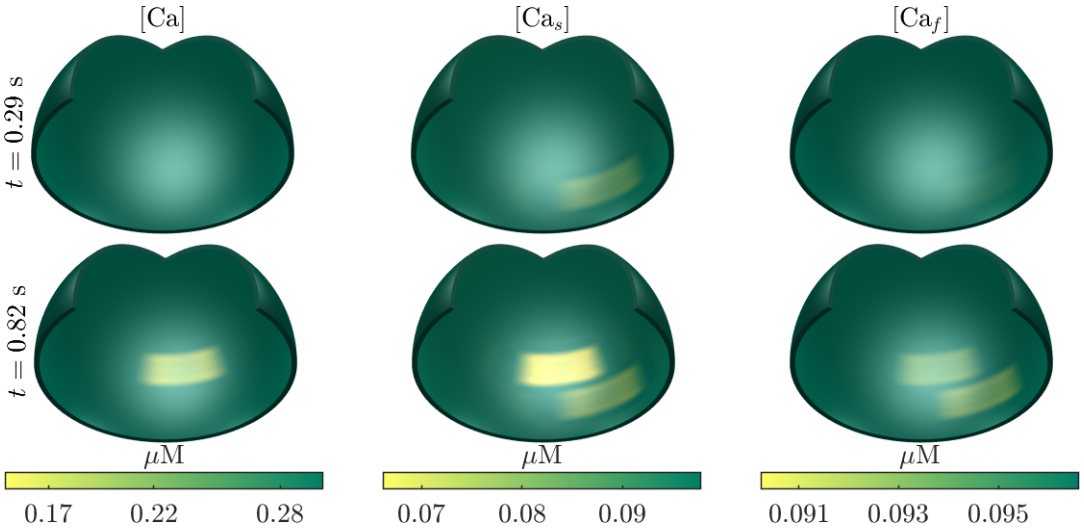


Figure 6.3: Top: The concentration of the outer segment calcium, [Ca], (left), inner segment submembrane calcium, [Ca$_s$], (center), inner segment central space calcium, [Ca$_f$], (right) of the L-cones domain at $t = 0.29$ s. The associated colour bar of each plot is located at the bottom of the plot's column. The inner segment calcium concentration is affected by light stimuli to another photoreceptor. As the movement of calcium is buffered, central space calcium response is not yet clearly visible. No outer segment response is observed. Bottom: similar to the top row but at $t = 0.82$ s. The newly observed response in all three plots is a result of a light stimuli to the L-cones domain, unlike the previous response, whose affects can still be observed in both inner segment calcium concentrations.

6.4 An Unhealthy Retina Affects ERG Measurements

As mentioned in Chapter 1, ERGs can be used to detect retinal diseases such as retinitis pigmentosa [26]. Using our model, we can study a variety of retinal diseases since we have extensive control over retinal cells in the model. For example, in numerical simulation 5, we use our model to study the effect of photoreceptor degeneration on ERG measurements. We do so by stimulating retinae with various patterns of photoreceptor degeneratrion (see Appendix A) with a spatially Gaussian ($\sigma = 50$), 20 ms light pulse.

Figure 6.4 shows a plot of the potential as a function of time at a point on the surface of the eye. As the plot shows, potential measurements at the surface of the eye, in other words ERG measurements, exhibit different features for the different retinae. An obvious distinguishing feature is the amplitude of the hyperpolarization response is diminished for the diseased retinae. The decrease in potential appears to be proportional to the extent of the damage of the retina. For instance, the most heavily damaged retina (with disease C) has a much weaker response amplitude than the other retinae. A more subtle distinguishing feature of the ERG measurements is that there appears to be a phase-shift in the slow depolarization phase between the various retinae. This is most apparent when comparing the retinae with disease A and B to the healthy one. It is the case that in real life such distinguishing features are used to diagnose diseases [26]. For example, ERGs can be used to identify individuals with myotonic dystrophies, a type of muscular dystrophy, even when the person does not exhibit any neurological symptoms [26]. This is because the ERG measurements of such individuals usually shows a significantly diminished b-wave [26].

It is also evident from Fig. 6.4 that this particular ERG measurement is not useful in distinguishing between retinae with disease A or B as the response is quite similar.

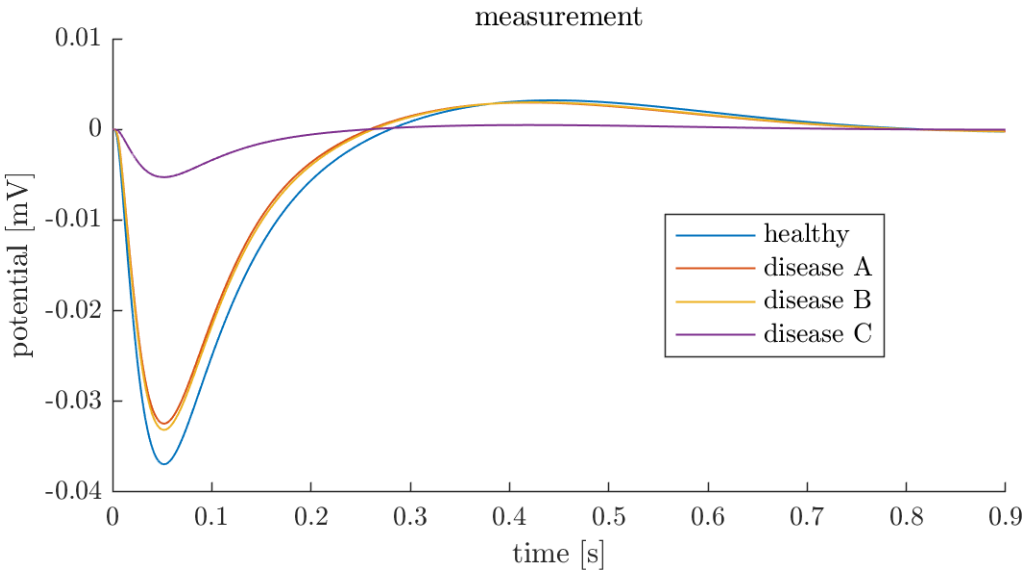


Figure 6.4: Extracellular potential at a point on the surface of the eye as a function of time for various retinae. See numerical simulation 5 in Appendix A for more details about the set-up of this simulation.

However, this does not imply that ERGs cannot be used to distinguish between the types of photoreceptor degeneration patterns. Perhaps a narrower light spot that is slightly off-center may evoke a more distinguished ERG response between the two types of retinae. It is also the case in the clinic that some ERG tests appear normal while others indicate the presence of a disease [26, 42]. For example, patients with congenital achromatopsia, sometimes referred to as colour blindness, can show normal rod responses while their cone responses are completely diminished [40, 42, 81].

In addition to controlling the light stimuli, we are also able to control the nature of the modelled retinal disease. For example, the results shown in Fig. 6.4 are based on diseased where the photoreceptor, L-cones in this case, is completely insensitive to light. For example, the photoreceptor is no longer able to manufacture the pigment that absorbs photons and starts the phototransduction pathway [44, 72]. Since we are using a biological model of the photoreceptor, we are able to, for example, interrupt the phototransduction pathway at a different stages instead. This shows the possibility that this model can be used for ERG diagnosis and potentially tracking the progression of a variety of diseases.

Chapter 7

Conclusion

7.1 Summary

In this dissertation, we presented a detailed model and simulation of the retina, which takes into account the retinal physiology as well as the geometry of the eye. The model is based on the bi-domain equations and is the first bi-domain retina model which takes into account the entirety of the eye. It is a versatile model in the sense that it can be used with any model of transmembrane currents. This model can be viewed as complimentary to the current retinal models, such as the one proposed by Dokos et al. [34], which were mainly concerned with electrode stimulation of and signal propagation through the retina. We also generalized our bi-domain model to a multi-domain model, which can account for all types of known photoreceptors. This significantly extends the scope of the model and increases the number of questions that can be investigated with it. We demonstrated some of these questions by presenting numerical simulations that compare favourably with experimental findings.

Furthermore, we detailed how we discretize the model's system of PDEs and our implicit time-stepping scheme which used BDF2 and a Newton-iterative method. We demonstrated how our adaptive time-stepper was used to significantly decrease

simulation time while maintaining accuracy. We also investigated various iterative methods for solving the arising Jacobian update equation in the Newton iteration and demonstrated how they can be used to significantly decrease simulation time while maintaining accuracy. While all this numerical work was in the context of our model, it is clearly applicable to other systems, most notably those arising from discretizing PDE systems with elliptic constraints. We then fully generalized our adaptive time-stepper to higher order BDFs, providing the coefficients and error bounds needed to implement the adaptive time-stepper. This further extends the use of the time-stepper to problems that cannot be satisfactorily solved with lower order methods.

7.2 Future Directions

There are three main distinct, yet interdependent, future directions for the presented work. Namely, improving the level of detail of the model, improving the numerics and implementation of the simulation, and working on applications of the model and simulation.

Currently, this model only accounts for one type of neurons present in the retina, namely the photoreceptors. Adding other types of neurons will increase the scope of applicability of this model since it will reproduce more features of the retina (for example, the b-wave mentioned in Section 6.1). However, this evidently will lead to higher computational cost of the simulation, which in turn places more importance on the numerics and implementation of the simulation.

On that front, a limitation of the model is the computational cost of this simulation. This, unfortunately, has been the downfall of cardiac tissue models using the bi-domain equations as well, which limited their usage [66]. Even though we were able to significantly cut this cost by the use of an adaptive time-stepper and iterative methods, there is still much left to be done. For example, the curse of dimensionality

is evident in our simulation. We tried to mitigate some the cost by using a non-uniform tensor product grid (see Section 4.1), however, that approach is limited. We hypothesize that implementing something like an algebraic multigrid method may significantly reduce computation cost [104]. A more minor change than multigrid methods is further investigation into the GMRES preconditioners. There has been many instances where preconditioners built based on information of the original problem out performed general ones [104, 51, 103, 125, 114]. While our reordering and scaling preconditioner may be categorized as such, the information used to built them is limited to the structure and number of diagonally dominant rows of the Jacobian (see Section 4.3.2). We hypothesize that using information about the physical and chemical properties of the underlying biological model will yield significantly better results.

Improving the numerics of the simulation will allow us to investigate applications of the model more effectively. For instance, an exciting area of application for this model is in aiding electroretinogram (ERG) diagnostics. The model can be manipulated to mimic many diseases that ERGs are used to diagnose. Thus, through various techniques such as parameter fitting, we hope to be able to use this model to replicate ERG measurements and subsequently aid ERG diagnostics. This will require numerous simulations of the model further emphasizing the importance of improving the numerics and implementation of the simulation. Another important application of this model is in providing deeper understanding of ERG measurements. For example, it has been shown that there is a correlation between ERG measurements and the thickness of a diseased retina [117, 16, 90, 91]. Improving the model to account for various geometries, such as variable thickness, of the retina will enable us to propose and understand mechanisms for such observed phenomena. This feeds into our first goal of improving the level of detail of the model.

Appendix A

Simulation Details

Numerical Simulation 1:

- Grid size ($r \times \theta \times \varphi$): $30 \times 29 \times 27$.
- Total number of unknowns: $142,352$.
- Simulation interval: $[0, 1]$ (seconds).
- Active Domains: L-cones.
- Stimulus: Spatially Gaussian ($\sigma = 50$), 1 s light flash at $t = 0$ s aimed at center of retina.

Numerical Simulation 2:

- Grid size ($r \times \theta \times \varphi$): $30 \times 29 \times 27$.
- Total number of unknowns: $142,352$.
- Simulation interval: $[0, 5]$ (seconds).
- Active Domains: L-cones.

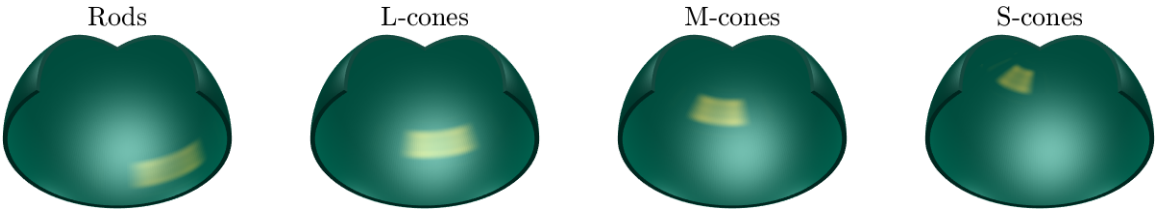


Figure A.1: Left to right: Location of light stimuli flashed at rods, L-cones, M-cones, and S-cones, respectively. The times of the 20 ms light pulses were at $t = 0, 0.5, 1.0, 1.5$ s for rods, L-cones, M-cones, and S-cones, respectively

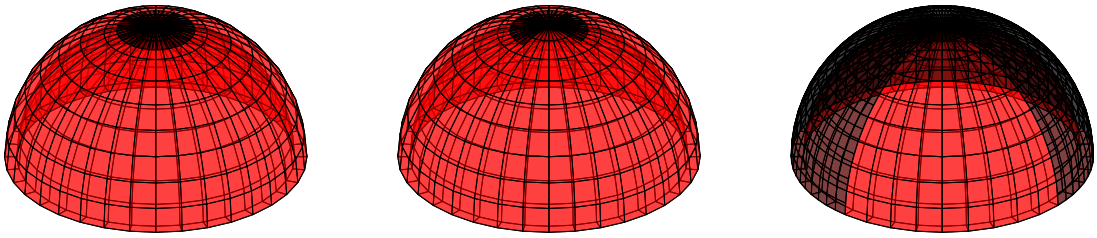
- Stimulus: Spatially Gaussian ($\sigma = 50$), 20 ms light pulse at $t = 0$ s aimed at center of retina.

Numerical Simulation 3:

- Grid size ($r \times \theta \times \varphi$): $30 \times 29 \times 27$.
- Total number of unknowns: $142,352$.
- Simulation interval: $[0, 5]$ (seconds).
- Active Domains: L-cones.
- Stimulus: Spatially Gaussian ($\sigma = 50$), 20 ms light pulses at $t = 0, 2.0, 2.1, 2.2, \ldots, 2.9$ s aimed at center of retina.

Numerical Simulation 4:

- Grid size ($r \times \theta \times \varphi$): $30 \times 35 \times 46$.
- Total number of unknowns: $1,146,412$.
- Simulation interval: $[0, 5]$ (seconds).
- Active Domains: Rods, L-cones, M-cones, and S-cones.
- Stimulus: Spatially-disjoint, 20 ms light pulses were given at $t = 0, 0.5, 1.0, 1.5$ s to rods, L-cones, M-cones, and S-cones, respectively (see Fig. A.1).



disease A disease B disease C

Figure A.2: Left to right: the different patterns of photoreceptor degeneration for disease A, disease B, and disease C retinae, respectively.

Numerical Simulation 5:

- Grid size ($r \times \theta \times \varphi$): $30 \times 29 \times 27$.
- Total number of unknowns: $142,352$.
- Simulation interval: $[0, 5]$ (seconds).
- Active Domains: L-cones.
- Stimulus: Spatially Gaussian ($\sigma = 50$), 20 ms light pulse at $t = 0$ s aimed at the center of a healthy retina and several retinae suffering from phtoreceptor degeneration (see Fig. A.2).

Appendix B

Supplementary Equations

Spherical Coordinate Representations of (2.3) to (2.5):

The spherical representation of (2.3) to (2.5) are

$$\frac{\partial \mu_{\mathrm{s}}^{r}}{\partial r}\frac{\partial \phi_{\mathrm{s}}}{\partial r}+\mu_{\mathrm{s}}^{r}\frac{\partial^{2}\phi_{\mathrm{s}}}{\partial r^{2}}+2\frac{\mu_{\mathrm{s}}^{r}}{r}\frac{\partial \phi_{\mathrm{s}}}{\partial r}+\frac{1}{r^{2}\cos^{2}\varphi}\left(\frac{\partial \mu_{\mathrm{s}}^{\theta}}{\partial \theta}\frac{\partial \phi_{\mathrm{s}}}{\partial \theta}+\mu_{\mathrm{s}}^{\theta}\frac{\partial^{2}\phi_{\mathrm{s}}}{\partial \theta^{2}}\right) - \frac{\mu_{\mathrm{s}}^{\varphi}\sin\varphi}{r^{2}\cos\varphi}\frac{\partial \phi_{\mathrm{s}}}{\partial \varphi}+\frac{1}{r^{2}}\left(\frac{\partial \mu_{\mathrm{s}}^{\varphi}}{\partial \varphi}\frac{\partial \phi_{\mathrm{s}}}{\partial \varphi}+\mu_{\mathrm{s}}^{\varphi}\frac{\partial^{2}\phi_{\mathrm{s}}}{\partial \varphi^{2}}\right)=0,$$

$$\begin{aligned}&\left[\frac{\partial \mu_{\mathrm{i}}^{r}}{\partial r}\frac{\partial V_{\mathrm{m}}}{\partial r}+\mu_{\mathrm{i}}^{r}\frac{\partial^{2}V_{\mathrm{m}}}{\partial r^{2}}+2\frac{\mu_{\mathrm{i}}^{r}}{r}\frac{\partial V_{\mathrm{m}}}{\partial r}+\frac{1}{r^{2}\cos^{2}\varphi}\left(\frac{\partial \mu_{\mathrm{i}}^{\theta}}{\partial \theta}\frac{\partial V_{\mathrm{m}}}{\partial \theta}+\mu_{\mathrm{i}}^{\theta}\frac{\partial^{2}V_{\mathrm{m}}}{\partial \theta^{2}}\right)\right.\\&\qquad\left.-\frac{\mu_{\mathrm{i}}^{\varphi}\sin\varphi}{r^{2}\cos\varphi}\frac{\partial V_{\mathrm{m}}}{\partial \varphi}+\frac{1}{r^{2}}\left(\frac{\partial \mu_{\mathrm{i}}^{\varphi}}{\partial \varphi}\frac{\partial V_{\mathrm{m}}}{\partial \varphi}+\mu_{\mathrm{i}}^{\varphi}\frac{\partial^{2}V_{\mathrm{m}}}{\partial \varphi^{2}}\right)\right]\\&+\left[\frac{\partial(\mu_{\mathrm{i}}^{r}+\mu_{\mathrm{e}}^{r})}{\partial r}\frac{\partial \phi_{\mathrm{e}}}{\partial r}+(\mu_{\mathrm{i}}^{r}+\mu_{\mathrm{e}}^{r})\frac{\partial^{2}\phi_{\mathrm{e}}}{\partial r^{2}}+2\frac{(\mu_{\mathrm{i}}^{r}+\mu_{\mathrm{e}}^{r})}{r}\frac{\partial \phi_{\mathrm{e}}}{\partial r}\right.\\&\quad+\frac{1}{r^{2}\cos^{2}\varphi}\left(\frac{\partial\big(\mu_{\mathrm{i}}^{\theta}+\mu_{\mathrm{e}}^{\theta}\big)}{\partial \theta}\frac{\partial \phi_{\mathrm{e}}}{\partial \theta}+\big(\mu_{\mathrm{i}}^{\theta}+\mu_{\mathrm{e}}^{\theta}\big)\frac{\partial^{2}\phi_{\mathrm{e}}}{\partial \theta^{2}}\right)-\frac{(\mu_{\mathrm{i}}^{\varphi}+\mu_{\mathrm{e}}^{\varphi})\sin\varphi}{r^{2}\cos\varphi}\frac{\partial \phi_{\mathrm{e}}}{\partial \varphi}\\&\qquad\left.+\frac{1}{r^{2}}\left(\frac{\partial(\mu_{\mathrm{i}}^{\varphi}+\mu_{\mathrm{e}}^{\varphi})}{\partial \varphi}\frac{\partial \phi_{\mathrm{e}}}{\partial \varphi}+(\mu_{\mathrm{i}}^{\varphi}+\mu_{\mathrm{e}}^{\varphi})\frac{\partial^{2}\phi_{\mathrm{e}}}{\partial \varphi^{2}}\right)\right]=0,\end{aligned}$$

$$\frac{\partial V_{\mathrm{m}}}{\partial t} = -\frac{\chi}{C_{\mathrm{m}}}\left[\frac{\partial \mu_{\mathrm{e}}^{r}}{\partial r}\frac{\partial \phi_{\mathrm{e}}}{\partial r} + \mu_{\mathrm{e}}^{r}\frac{\partial^2 \phi_{\mathrm{e}}}{\partial r^2} + 2\frac{\mu_{\mathrm{e}}^{r}}{r}\frac{\partial \phi_{\mathrm{e}}}{\partial r} + \frac{1}{r^2\cos^2\varphi}\left(\frac{\partial \mu_{\mathrm{e}}^{\theta}}{\partial \theta}\frac{\partial \phi_{\mathrm{e}}}{\partial \theta} + \mu_{\mathrm{e}}^{\theta}\frac{\partial^2 \phi_{\mathrm{e}}}{\partial \theta^2}\right)\right.$$
$$\left. - \frac{\mu_{\mathrm{e}}^{\varphi}\sin\varphi}{r^2\cos\varphi}\frac{\partial \phi_{\mathrm{e}}}{\partial \varphi} + \frac{1}{r^2}\left(\frac{\partial \mu_{\mathrm{e}}^{\varphi}}{\partial \varphi}\frac{\partial \phi_{\mathrm{e}}}{\partial \varphi} + \mu_{\mathrm{e}}^{\varphi}\frac{\partial^2 \phi_{\mathrm{e}}}{\partial \varphi^2}\right)\right] - \frac{1}{C_{\mathrm{m}}} I_{\mathrm{m}}(V_{\mathrm{m}}, \boldsymbol{X}),$$

respectively.

Finite Difference Equations for all variables: Let $\{r^0 = 0, \ldots, r^n = r_{\mathrm{eye}}\}, \{\theta^0 = 0, \ldots, \theta^m\}, \{\varphi^0 > \pi/2, \ldots, \varphi^p < \pi/2\}$ be the discretization points in the radial direction, polar, and latitudinal directions, respectively. Also let $\phi^{(i,j,k)}$ correspond to the value of ϕ at the (i, j, k) node, where i, j, k are indices for the radial, polar, and latitudinal directions, respectively.

$$\begin{aligned}
\frac{\partial \phi^{(i,j,k)}}{\partial r} &= \frac{\phi^{(i+1,j,k)} - \gamma_i^2\phi^{(i-1,j,k)} - (1-\gamma_i^2)\phi^{(i,j,k)}}{(1+\gamma_i)h_{i+1}},\\
\frac{\partial^2 \phi^{(i,j,k)}}{\partial r^2} &= 2\gamma_i\frac{\phi^{(i+1,j,k)} + \gamma_i\phi^{(i-1,j,k)} - (1+\gamma_i)\phi^{(i,j,k)}}{(1+\gamma_i)h_{i+1}^2},\\
\frac{\partial \phi^{(i,j,k)}}{\partial \theta} &= \frac{\phi^{(i,j+1,k)} - \gamma_i^2\phi^{(i,j-1,k)} - (1-\gamma_i^2)\phi^{(i,j,k)}}{(1+\gamma_i)h_{i+1}},\\
\frac{\partial^2 \phi^{(i,j,k)}}{\partial \theta^2} &= 2\gamma_i\frac{\phi^{(i,j+1,k)} + \gamma_i\phi^{(i,j-1,k)} - (1+\gamma_i)\phi^{(i,j,k)}}{(1+\gamma_i)h_{i+1}^2},\\
\frac{\partial \phi^{(i,j,k)}}{\partial r} &= \frac{\phi^{(i,j,k+1)} - \gamma_i^2\phi^{(i,j,k-1)} - (1-\gamma_i^2)\phi^{(i,j,k)}}{(1+\gamma_i)h_{i+1}},\\
\frac{\partial^2 \phi^{(i,j,k)}}{\partial r^2} &= 2\gamma_i\frac{\phi^{(i,j,k+1)} + \gamma_i\phi^{(i,j,k-1)} - (1+\gamma_i)\phi^{(i,j,k)}}{(1+\gamma_i)h_{i+1}^2},
\end{aligned}$$

in which $h_{i+1} = x^{i+1} - x^i$ ($x \in \{r, \theta, \varphi\}$ is the obvious choice of variable) and $\gamma_i = \frac{h_{i+1}}{h_i}$.

Bibliography

[1] A. A. Abed, N. H. Lovell, G. J. Suaning, and S. Dokos, *A hybrid continuum-discrete computational model of electrical stimulation of the retinal network*, 2015 7th International IEEE/EMBS Conference on Neural Engineering (NER), (2015), pp. 352–355.

[2] M. Abramian, N. H. Lovell, J. W. Morley, G. J. Suaning, and S. Dokos, *Activation of retinal ganglion cells following epiretinal electrical stimulation with hexagonally arranged bipolar electrodes*, Journal of Neural Engineering, 8 (2011), p. 035004.

[3] B. Abuelnasr and A. R. Stinchcombe, *A multi-scale simulation of retinal physiology*, Mathematical Biosciences, 363 (2023), p. 109053.

[4] A. Alqahtani, A. A. Abed, E. E. Anderson, N. H. Lovell, and S. Dokos, *A multi-domain continuum model of electrical stimulation of healthy and degenerate retina*, 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), (2018), pp. 6117–6120.

[5] A. Alqahtani, A. A. Abed, T. Guo, N. H. Lovell, and S. Dokos, *A continuum model of electrical stimulation of multi-compartmental retinal ganglion cells*, 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2017 (2017), pp. 2716–2719.

[6] A. Alqahtani, A. A. Abed, N. H. Lovell, and S. Dokos, *Optimizing stimulation strategies for retinal electrical stimulation: a modelling study*, 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), (2019), pp. 2872–2875.

[7] T. Aoyama, Y. Kamiyama, S. Usui, R. Blanco, C. F. Vaquero, and P. de la Villa, *Ionic current model of rabbit retinal horizontal cell*, Neuroscience Research, 37 (2000), pp. 141–151.

[8] W. E. Arnoldi, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quarterly of Applied Mathematics, 9 (1951), pp. 17–29.

[9] U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, Society of Industrial and Applied Mathematics, 1998.

[10] S. Barnes and B. Hille, *Ionic channels of the inner segment of tiger salamander cone photoreceptors*, The Journal of General Physiology, 94 (1989).

[11] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. A. van der Vorst, *Templates for The Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, 1994.

[12] J. A. Bennett and B. J. Roth, *Time dependence of anodal and cathodal refractory periods in cardiac tissue*, Pacing and Clinical Electrophysiology, 22 (1999), pp. 1031–1038.

[13] M. Benzi, *Preconditioning techniques for large linear systems: A survey*, Journal of Computational Physics, 182 (2002), pp. 418–477.

[14] M. Benzi, J. C. Haws, and M. Tuma, *Preconditioning highly indefinite and nonsymmetric matrices*, SIAM Journal on Scientific Computing, 22 (2000), pp. 1333–1353.

[15] M. Benzi, D. B. Szyld, and A. V. Duin, *Orderings for incomplete factorization preconditioning of nonsymmetric problems*, SIAM Journal on Scientific Computing, 20 (1999), pp. 1652–1670.

[16] D. G. Birch, P. D. Williams, D. Callanan, R. Wang, K. G. Locke, and D. C. Hood, *Macular atrophy in birdshot retinochoroidopathy: An optical coherence tomography and multifocal electroretinography analysis*, Retina, 30 (2010), pp. 930–937.

[17] E. Bodewig, *Matrix Calculus*, North-Holland Publishing Company, 2nd edition ed., 1959.

[18] P. N. Brown and H. F. Walker, *GMRES on (nearly) singular systems*, SIAM Journal on Matrix Analysis and Applications, 18 (1997), pp. 37–51.

[19] D. Calvetti, B. Lewis, and L. Reichel, *GMRES-type methods for inconsistent systems*, Linear Algebra and its Applications, 316 (2000), pp. 157–169.

[20] S. L. Campbell, I. C. F. Ipsen, C. T. Kelley, and C. D. Meyer, *GMRES and the minimal polynomial*, BIT Numerical Mathematics, 36 (1996), pp. 664–675.

[21] Z.-H. Cao and M. Wang, *A note on Krylov subspace methods for singular systems*, Linear Algebra and its Applications, 350 (2002), pp. 285–288.

[22] K. Chen, *Matrix Preconditioning Techniques and Applications*, Cambridge Monographs on Applied Computational Mathematics, Cambridge University Press, 2005.

[23] D. Cioranescu and P. Donato, *An Introduction to Homogenization*, Oxford University Press, 1999.

[24] A. I. Cohen, *Interphotoreceptor contacts at the inner segment level in primate retinas*, Brain Research, 490 (1989), pp. 200–203.

[25] E. D. Cohen, *Retinal prostheses*, in Webvision: The Organization of the Retina Visual System, H. Kolb, E. Fernandez, and R. Nelson, eds., University of Utah Health Sciences Center, 2018, pp. 1607–1629.

[26] D. J. Creel, *Clinical electrophysiology*, in Webvision: The Organization of the Retina Visual System, H. Kolb, E. Fernandez, and R. Nelson, eds., University of Utah Health Sciences Center, 2007, pp. 1235–1290.

[27] A. R. Curtis and J. K. Reid, *On the automatic scaling of matrices for Gaussian elimination*, Journal of the Institute of Mathematics and its Applications, 10 (1972), pp. 118–124.

[28] E. Cuthill, *Several strategies for reducing the bandwidth of matrices*, in Sparse Matrices and their Applications, D. J. Rose and R. A. Willoughby, eds., Springer, 1972, pp. 157–166.

[29] E. Cuthill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices*, in Proceedings of the 1969 24th National Conference, Association for Computing Machinery, 1969, pp. 157–172.

[30] E. Dekker, *Direct current make and break thresholds for pacemaker electrodes on the canine ventricle*, Circulation Research, 27 (1970), pp. 811–823.

[31] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 400–408.

[32] E. G. D'Jakonov, *An iteration method for solving systems of finite difference equations*, Dokl. Akad. Nauk SSSR, 138 (1961).

[33] ——, *On certain iterative methods for solving nonlinear difference equations*, in Conference on the Numerical Solution of Differential Equations, J. L. Morris, ed., Springer Berlin Heidelberg, 1969, pp. 7–22.

[34] S. Dokos, G. J. Suaning, and N. H. Lovell, *A bidomain model of epiretinal stimulation*, IEEE Transactions on Neural Systems and Rehabilitation Engineering, 13 (2005), pp. 137–146.

[35] J. E. Dowling, *The Retina: An Approachable Part of the Brain*, Belknap Press, revised edition ed., 2012.

[36] I. S. Duff and J. Koster, *The design and use of algorithms for permuting large entries to the diagonal of sparse matrices*, SIAM Journal on Matrix Analysis and Applications, 20 (1999), pp. 889–901.

[37] ——, *On algorithms for permuting large entries to the diagonal of a sparse matrix*, SIAM Journal on Matrix Analysis and Applications, 22 (2001), pp. 973–996.

[38] M. Eiermann, O. G. Ernst, and O. Schneider, *Analysis of acceleration strategies for restarted minimal residual methods*, Journal of Computational and Applied Mathematics, 123 (2000), pp. 261–292.

[39] M. Embree, *The tortoise and the hare restart GMRES*, SIAM Review, 45 (2003), pp. 259–266.

[40] H. F. Falls, J. R. Wolter, and M. Alpern, *Typical total monochromacy: A histological and psychophysical study*, Archives of Ophthalmology, 74 (1965), pp. 610–616.

[41] E. Fernandez and R. Normann, *Introduction to visual prostheses*, in Webvision: The Organization of the Retina Visual System, H. Kolb, E. Fernandez, and R. Nelson, eds., University of Utah Health Sciences Center, 2016, pp. 1581–1606.

[42] G. A. Fishman, D. G. Birch, G. E. Holder, and M. G. Brigell, *Electrophysiologic Testing in Disorders of the Retina, Optic Nerve, and Visual Pathway*, Foundation of the American Academy of Ophthalmology, 2nd edition ed., 2001.

[43] V. M. Fridman, *The method of minimum iterations with minimum errors for a system of linear algebraic equations with a symmetrical matrix*, U.S.S.R Computational Mathematics and Mathematical Physics, 2 (1963), pp. 362–363.

[44] Y. Fu, *Phototransduction in rods and cones*, in Webvision: The Organization of the Retina Visual System, H. Kolb, E. Fernandez, and R. Nelson, eds., University of Utah Health Sciences Center, 2010, pp. 457–493.

[45] Y. Fu and K.-W. Yau, *Phototransduction in mouse rods and cones*, Pflügers Archiv - European Journal of Physiology, 454 (2007), pp. 805–819.

[46] C. W. Gear and L. R. Petzold, *ODE methods for the solution of differential/algebraic systems*, SIAM Journal on Numerical Analysis, 21 (1984), pp. 716–728.

[47] C. W. Gear and K. W. Tu, *The effect of variable mesh size on the stability of multistep methods*, SIAM Journal on Numerical Analysis, 11 (1974), pp. 1025–1043.

[48] C. W. Gear and D. S. Watanabe, *Stability and convergence of variable order multistep methods*, SIAM Journal on Numerical Analysis, 11 (1974), pp. 1044–1058.

[49] J. A. George, *Computer Implementation of the Finite Element Method*, PhD thesis, Stanford University, 1971.

[50] J. A. George and J. W.-H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, 1981.

[51] G. H. Golub and A. J. Wathen, *An iteration for indefinite systems and its application to the Navier–Stokes equations*, SIAM Journal on Scientific Computing, 19 (1998), pp. 530–539.

[52] R. D. Grigorieff, *Stability of multistep-methods on variable grids*, Numerische Mathematik, 42 (1983), pp. 359–377.

[53] E. Hairer, S. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, Springer, 2008.

[54] E. Hairer and G. Wanner, *On the instability of the BDF formulas*, SIAM Journal on Numerical Analysis, 20 (1983), pp. 1206–1209.

[55] C. S. Henriquez and W. Ying, *The bidomain model of cardiac tissue: From microscale to macroscale*, in Cardiac Bioelectric Therapy: Mechanism and Practical Implications, I. Efimov, F. S. Ng, and J. Laughner, eds., Springer, second edition ed., 2021, ch. 15, pp. 401–421.

[56] M. R. Hestenes, *Iterative methods for solving linear equations*, Journal of Optimization Theory and Applications, 11 (1973), pp. 323–334.

[57] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, Journal of Research of the National Bureau of Standards, 49 (1952), pp. 409–436.

[58] Q. V. Hoang, R. A. Linsenmeier, C. K. Chung, and C. A. Curcio, *Photoreceptor inner segments in monkey and human retina: Mitochondrial density, optics, and regional variation*, Visual Neuroscience, 19 (2002), pp. 395–407.

[59] B. M. Invergo, D. Dell'Orco, L. Montanucci, K.-W. Koch, and J. Bertranpetit, *A comprehensive model of the phototransduction cascade in mouse rod cells*, Molecular BioSystems, 10 (2014).

[60] I. C. F. Ipsen and C. D. Meyer, *The idea behind Krylov methods*, The American Mathematical Monthly, 105 (1998), pp. 889–899.

[61] C. G. J. Jacobi, *Ueber eine neue auflösungsart der bei der methode der kleinsten quadrate vorkommenden lineären gleichungen*, Astronomische Nachrichten, 22 (1845), pp. 297–306.

[62] S. A. Joarder, M. Abramian, G. J. Suaning, N. H. Lovell, and S. Dokos, *A continuum model of retinal electrical stimulation*, Journal of Neural Engineering, 8 (2011), p. 0666006.

[63] S. A. Joarder, S. Dokos, G. J. Suaning, and N. H. Lovell, *Finite element bidomain model of epiretinal stimulation*, 2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2007 (2007), pp. 1132–1135.

[64] Y. Kamiyama, T. Ogura, and S. Usui, *Ionic current model of the vertebrate rod photoreceptor*, Vision Research, (1996).

[65] S. M. Kandel and B. J. Roth, *The strength-interval curve in cardiac tissue*, Computational and Mathematical Methods in Medicine, 2013 (2013).

[66] J. Keener and J. Sneyd, *Mathematical Physiology*, Springer, 2009.

[67] D. Kincaid and W. Cheney, *Numerical Analysis: Mathematics of Scientific Computing*, vol. 2 of Pure and Applied Undergraduate Texts, American Mathematical Society, third edition ed., 2002.

[68] S. B. Knisley, *Transmembrane voltage changes during unipolar stimulation of rabbit ventricle*, Circulation Research, 77 (1995), pp. 1229–1239.

[69] S. B. Knisley, B. C. Hill, and R. E. Ideker, *Virtual electrode effects in myocardial fibers*, Biophysical Journal, 66 (1994), pp. 719–728.

[70] H. Kolb, *The organization of the outer plexiform layer in the retina of the cat: Electron microscopic observation*, Journal of Neurocytology, 6 (1977), pp. 131–153.

[71] ——, *Gross anatomy of the eye*, in Webvision: The Organization of the Retina Visual System, H. Kolb, E. Fernandez, and R. Nelson, eds., University of Utah Health Sciences Center, 2005, pp. 3–10.

[72] ——, *Photoreceptors*, in Webvision: The Organization of the Retina Visual System, H. Kolb, E. Fernandez, and R. Nelson, eds., University of Utah Health Sciences Center, 2005, pp. 59–89.

[73] ——, *Simple anatomy of the retina*, in Webvision: The Organization of the Retina Visual System, H. Kolb, E. Fernandez, and R. Nelson, eds., University of Utah Health Sciences Center, 2005, pp. 11–34.

[74] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, Journal of Research of the National Bureau of Standards, 45 (1950), pp. 255–282.

[75] ——, *Solution of systems of linear equations by minimized iterations*, Journal of Research of the National Bureau of Standards, 49 (1952), pp. 33–53.

[76] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*, Society of Industrial and Applied Mathematics, 2007.

[77] Z. Liao, K. Hayami, K. Morikuni, and J.-F. Yin, *A stabilized GMRES method for singular and severely ill-conditioned systems of linear equations*, Japan Journal of Industrial and Applied Mathematics, 39 (2022), pp. 717–751.

[78] W.-H. Liu and A. H. Sherman, *Comparative analysis of the Cuthill-Mckee and the reverse Cuthill-Mckee ordering algorithms for sparse matrices*, SIAM Journal on Numerical Analysis, 13 (1976), pp. 198–213.

[79] D. G. Luenberger, *Hyperbolic pairs in the method of conjugate gradients*, SIAM Journal on Applied Mathematics, 17 (1969), pp. 1263–1267.

[80] ——, *The conjugate residual method for constrained minimization problems*, SIAM Journal on Numerical Analysis, 7 (1970), pp. 390–398.

[81] P. H. Madsen, *Total achromatopsia in two brothers*, Acta Ophthalmologica, 45 (1967), pp. 587–593.

[82] P. A. McNaughton, L. Cervetto, and B. J. Nunn, *Measurement of the intracellular free calcium concentration in salamander rods*, Nature, 322 (1986), pp. 261–263.

[83] H. Meffin, B. Tahayori, E. N. Sergeev, I. M. Y. Mareels, D. B. Grayden, and A. N. Burkitt, *Modelling extracellular electrical stimulation: Part 3. derivation and interperatiuon of neural tissue equations*, Journal of Neural Engineering, 11 (2014), p. 065004.

[84] R. Mehra, M. McMullen, and S. Furman, *Time-dependence of unipolar cathodal and anodal strength-interval curves*, Pacing and Clinical Electrophysiology, 3 (1980), pp. 526–530.

[85] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric m-matrix*, Mathematics of Computation, 31 (1977), pp. 148–162.

[86] J. C. Meza, *A modification to the GMRES method for ill-conditioned linear systems*, Tech. Rep. SAND95-8220, Sandia National Laboratories, 1995.

[87] S. S. Nagarajan, D. M. Durand, B. J. Roth, and R. S. Wijesinghe, *Magnetic stimulation of axons in a nerve bundle: Effects of current redistribution in the bundle*, Annals of Biomedical Engineering, 23 (1995), pp. 116–126.

[88] M. Neunlist and L. Tung, *Optical recordings of ventricular excitability of frog heart by an extracellular stimulating point electrode*, Pacing and Clinical Electrophysiology, 17 (1994), pp. 1641–1654.

[89] ——, *Spatial distribution of cardiac transmembrane potentials around an extracellular electrode: Dependence on fiber orientation*, Biophysical Journal, 68 (1995), pp. 2310–2322.

[90] H. Noma, H. Funatsu, S. Harino, T. Sugawara, T. Mimura, and K. Shimada, *Association of electroretinogram and morphological findings in branch retinal vein occlusion with macular edema*, Documenta Ophthalmologica, 123 (2011), pp. 83–91.

[91] H. Noma, T. Mimura, M. Kuse, K. Yasuda, and M. Shimura, *Photopic negative response in branch retinal vein occlusion with macular edema*, International Ophthalmology, 35 (2015), pp. 19–26.

[92] M. Olschowka and A. Neumaier, *A new pivoting strategy for Gaussian elimination*, Linear Algebra and its Applications, 240 (1996).

[93] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM Journal on Numerical Analysis, 12 (1975), pp. 617–629.

[94] I. Perlman, *The electroretinogram: ERG*, in Webvision: The Organization of the Retina Visual System, H. Kolb, E. Fernandez, and R. Nelson, eds., University of Utah Health Sciences Center, 2001, pp. 1193–1234.

[95] E. Raviola and N. B. Gilula, *Gap junctions between photoreceptor cells in the vertebrate retina*, Proceedings of the National Academy of Sciences of the United States of America, 70 (1973), pp. 1677–1681.

[96] L. Reichel and Q. Ye, *Breakdown-free GMRES for singular systems*, SIAM Journal on Matrix Analysis and Applications, 26 (2005), pp. 1001–1021.

[97] L. F. Richardson, *The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam*, Philosphical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character, 210 (1911), pp. 307–357.

[98] R. W. Rodieck, *The First Steps in Seeing*, Sinauer, 1998.

[99] B. J. Roth, *How the anisotrpoy of the intracellular and extracellular conductivities influences stimulation of cardiac muscle*, Journal of Mathematical Biology, 30 (1992), pp. 633–646.

[100] ——, *A mathematical model of make and break electrical stimulation of cardiac tissue by a unipolar anode or cathode*, IEEE Transactions on Biomedical Engineering, 42 (1995), pp. 1174–1184.

[101] ——, *Strength-interval curves for cardiac tissue predicted using the bidomain model*, Journal of cardiovascular electrophysiology, 7 (1996), pp. 722–737.

[102] B. J. Roth and J. P. J. Wikswo, *Electrical stimulation of cardiac tissue: a bidomain model with active membrane properties*, IEEE Transactions on Biomedical Engineering, 41 (1994), pp. 232–240.

[103] T. Rusten and R. Winther, *A preconditioned iterative method for saddle-point problems*, SIAM Journal on Matrix Analysis and Applications, 13 (1992), pp. 887–904.

[104] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 2nd edition ed., 2003.

[105] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 7 (1986), pp. 856–869.

[106] R. J. Sadleir, *A bidomain model for neural tissue*, International Journal of Bioelectromagnetism, 12 (2010), pp. 2–6.

[107] R. J. Sadleir, F. Fu, and M. Chauhan, *Functional magnetic resonance electrical impedance tomography (fMREIT) sensitivity analysis using an active bidomain finite-element model of neural tissue*, Magnetic Resonance in Medicine, 81 (2019).

[108] M. Salahi, *Nonnegative ill-conditioned linear systems and GMRES method*, Journal of Applied Mathematics & Computing, 31 (2009), pp. 507–515.

[109] N. G. Sepulveda, B. J. Roth, and J. P. J. Wikswo, *Current injection into a two-dimensional anisotropic bidomain*, Biophysical Journal, 55 (1989), pp. 987–999.

[110] F. Shalbaf, S. Dokos, N. H. Lovell, J. Turuwhenua, and E. Vaghefi, *Generating 3d anatomically detailed models of the retina from oct data sets: Implications for computational modelling*, Journal of Modern Optics, 62 (2015), pp. 1789–1800.

[111] F. Shalbaf, P. Du, N. H. Lovell, and S. Dokos, *A 3d-continuum bidomain model of retinal electrical stimulation using an anatomically detailed mesh*, 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2015 (2015), pp. 2291–2294.

[112] F. Shalbaf, N. H. Lovell, S. Dokos, M. Trew, and E. Vaghefi, *Foveal eccentricity can influence activation threshold in subretinal electrical simulation*, Biomedical Physics and Engineering Express, 5 (2019), p. 35009.

[113] A. H. Sherman, *On Newton-iterative methods for the solution of systems of nonlinear equations*, SIAM Journal on Numerical Analysis, 15 (1978), pp. 755–771.

[114] D. Silvester and A. Wathen, *Fast iterative solution of stabilised stokes systems part ii: Using general block preconditioners*, SIAM Journal on Numerical Analysis, 31 (1994), pp. 1352–1367.

[115] R. G. Smith, M. A. Freed, and P. Sterling, *Microcircuitry of the dark-adapted cat retina: Functional architecture of the rod-cone network*, The Journal of Neuroscience, 6 (1986), pp. 3505–3517.

[116] T. Steihaug, *Quasi-Newton Methods for Large-Scale Nonlinear Problems*, PhD thesis, Yale University, 1981.

[117] H. Terasaki, K. Miyake, and Y. Miyake, *Reduced oscillatory potentials of the full-field electroretinogram of eyes with aphakic or pseudophakic cystoid macular edema*, American Journal of Opthalmology, 135 (2003), pp. 477–482.

[118] V. Torre, S. Forti, A. Menini, and M. Campani, *Model of phototransduction in retinal rods*, Cold Spring Harbor Symposia on Qunatitative Biology, LV (1990), pp. 563–573.

[119] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, 1997.

[120] L. Tung, *A Bi-domain Model for Describing Ischemic Myocardial D-C Potentials*, PhD thesis, Massachusetts Institute of Technology, 1978.

[121] L. Tung, M. Neunlist, and E. A. Sobie, *Near-field and far-field stimulation of cardiac muscle*, in Clinical Applications of Modern Imaging Technology II, L. J. Cerullo, K. S. Heiferman, H. Liu, H. Podbielska, A. O. Wist, and L. J. Zamorano, eds., vol. 2132, International Society for Optics and Photonics, SPIE, 1994, pp. 367–374.

[122] A. M. Turing, *Rounding-off errors in matrix processes*, Quarterly Journal of Mechanics and Applied Mathematics, 1 (1948), pp. 287–308.

[123] S. Usui, A. Ishihara, Y. Kamiyama, and H. Ishii, *Ionic current model of bipolar cells in the lower vertebrate retina*, Vision Research, 36 (1996), pp. 4069–4076.

[124] S. Usui, Y. Kamiyama, H. Ishii, and H. Ikeno, *Reconstruction of retinal horizontal cell responses by the ionic current model*, Vision Research, 36 (1996), pp. 1711–1719.

[125] A. Wathen and D. Silvester, *Fast iterative solution of stabilised stokes systems part i: Using simple diagonal preconditioners*, SIAM Journal on Numerical Analysis, 30 (1993), pp. 630–649.

[126] J. Weiland and M. Humayun, *Retinal prosthesis*, in Neural Engineering, B. He, ed., Springer, 2nd edition ed., 2013, pp. 635–655.

[127] K. C. Wikler and P. Rakic, *Distribution of photoreceptor subtypes in the retina of diurnal and nocturnal primates*, The Journal of Neuroscience, 10 (1990), pp. 3390–3401.

[128] J. P. J. Wikswo, S.-F. Lin, and R. A. Abbas, *Virtual electrodes in cardiac tissue: A common mechanism for anodal and cathodal stimulation*, Biophysical Journal, 69 (1995), pp. 2195–2210.

[129] J. P. J. Wikswo, T. A. Wisialowski, W. A. Altemeier, J. R. Balser, H. A. Kopelman, and D. M. Roden, *Virtual cathode effects during stimulation of cardiac muscle: Two-dimensional in vivo experiments*, Circulation Research, 68 (1991), pp. 513–530.

[130] D. Yan, M. C. Pugh, and F. P. Dawson, *Adaptive time-stepping schemes for the solution of the Poisson-Nernst-Planck equations*, Applied Numerical Mathematics, 163 (2021), pp. 254–269.

[131] S. Yin, N. H. Lovell, G. J. Suaning, and S. Dokos, *A continuum model of the retinal network and its response to electrical stimulation*, 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology, 2010 (2010), pp. 2077–2080.

[132] ——, *Continuum model of light response in the retina*, 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2011 (2011).

[133] D. M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, 1971.

[134] J. Zang and S. C. F. Neuhauss, *Biochemistry and physiology of zebrafish photoreceptors*, Pflügers Archiv - European Journal of Physiology, 473 (2021), pp. 1569–1585.