MODELLING THE SPATIAL EFFECTS OF THE SIGNAL TRANSDUCTION PROCESS

by

Chris Levy

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
April 2015

Table of Contents

List of Tables

List of Figures

Abstract

In this thesis we construct and analyze a cell signal transduction model for a biological cell which takes into account three dimensional geometry, diffusion, and spatial separation and localization of activating and deactivating enzymes. The deactivation of signalling proteins occurs throughout the cytosol and activation is localized to specific sites in the cell. The model consists of a system of linear reaction diffusion equations (PDEs) and nonlinear boundary conditions defined over a spherical domain with spherical compartments within its interior. Using asymptotic methods we obtain steady state solutions and determine their stability. We also find asymptotic time dependent solutions to the PDE model. We do this analysis for a model with and without delay. The delay is used to model the time lapse during enzyme reactions and also the recovery times associated with conformational changes during the phosphorylation process. For the model without delay, the full PDE system is approximated by a system of differential algebraic equations. The full model with delay is approximated by a system of delay differential algebraic equations. From our results we can detect complex signalling behavior such as bistability, robust switches, and sustained oscillations which all come about from different bifurcations. The simulations of the full three dimensional systems correspond well with simulations of the approximating differential algebraic systems. In this thesis we also introduce a model with cell surface receptors. We model the clustering of these receptors by introducing circular patches on the surface of the sphere. We then investigate the effect that the number of clusters has on the signalling pathway.

List of Abbreviations and Symbols Used

ODE ..... ordinary differential equation

BVP ..... boundary value problem

PDE ..... partial differential equation

DDE ..... delay differential equation

DAE ..... differential algebraic equation

DDAE ..... delay differential algebraic equation

GK ..... Goldbeter–Koshland

MAPK ..... mitogen-activated protein kinase

ε ..... radius of signalling compartment

$O(\varepsilon)$ ..... Landau notation

D_u ..... diffusion coefficient

k_u ..... decay rate

R ..... radius of the cell

λ ..... eigenvalue

α_u ..... $\alpha_u = \sqrt{k_u/D_u}$, measures diffusion length in terms of cell radius

Ω_1 ..... sphere of radius 1 centered at the origin

Ω_{ε_1} ..... sphere of radius ε centered at x_1

Ω_ε ..... union of two interior signalling compartments

$\Omega_1/\Omega_\varepsilon$ ..... cell region minus the two compartments

$\hat{n}$ ..... the normal vector

Δ_x ..... laplace operator in 3D cartesian coordinates

∂_{nx} ..... $\hat{n} \cdot \nabla_x$, the outward normal derivative

θ ..... azimuth coordinate in spherical coordinates

ϕ ..... zenith coordinate in spherical coordinates

ρ ..... $|y|$, the inner radial coordinate in spherical coordinates

Δ_y ..... $\partial_\rho^2 + (2/\rho)\partial_\rho$, spherically symmetric laplacian

Acknowledgements

First I would like to thank David Iron for being my supervisor. He was a great support and I would like to thank him for coming up with such a good problem to work on. I would also like to thank Paul Muir and Theodore Kolokolnikov for being on my supervisory committee and for reading my thesis. I am also grateful for the faculty and staff within the Mathematics and Statistics department who either taught me or helped me out with numerous things over the past several years. I would like to especially thank Sara Faridi for encouraging me to apply for a Killam graduate scholarship which funded the last two years of my research. I am also very thankful for NSERC which funded the first three years of my PhD. I also would like to say a special thank you to Balagopal Pillai. He was especially helpful in getting me connected to the cluster and keeping me updated on different versions of Comsol and Matlab which played a huge role in my research. Finally I would like to thank my wife Joanna for her support and for taking care of things at home with our kids.

Chapter 1

Introduction

Some of the descriptions and contents in this chapter have been published among two papers [61, 62] written by us. The results from [61], which motivate the work done in this thesis, are summarized in §1.2. The majority of Chapter 2 is published in [62]. The papers [61, 62] are published in the Journal of Math Biology and the final publications are available at http://link.springer.com. The material in Chapter 3 has been submitted to the Journal of Nonlinearity [63].

1.1 Signal Transduction Pathways and ODE Models

Biological cells in living organisms have the ability to respond to many different chemical stimuli within their environment. Cells receive signals and then carry out a response. Signal transduction pathways are the tools which cells use to receive and process stimuli. They are like molecular circuits which detect, amplify, and integrate external signals. The end result of signal transduction is a cellular response or decision. Examples of these responses are cytoskeletal reorganization, proliferation, differentiation, motility, cell survival, and cell death.

The malfunctioning of signalling pathways is a leading cause of human diseases such as cancer, chronic inflammatory syndromes and diabetes [49]. Qualitative and quantitative analysis of mathematical models helps in understanding the mechanisms that underlie the functions of signalling networks and therefore can supplement research on pharmacological interventions in the treatment of human diseases.

The signal transduction process begins when an extracellular signalling molecule binds to a cell surface receptor. In general, extracellular signalling molecules are quite large and do not diffuse across the cell membrane. Once the extracellular signalling molecule is bound to the cell surface receptor, the receptor performs the function of transferring the signal across the membrane and into the cell interior. The binding of the molecule with the receptor actually initiates a series of chemical

reactions inside the cell. These reactions carry the signal from the receptor bound signalling molecule, across the cell membrane, and through the cell interior. This then leads to a cellular response. Signal transduction is this process in which the signal travels from the cell surface bound receptor, across the cell membrane, and through the cell's cytoplasm (i.e. interior of the cell). This signalling pathway is often referred to as a signal transduction pathway (see Figure 1.1).

Cell surface receptors have both an intracellular and extracellular domain. The binding site on the extracellular portion of the receptor can recognize extracellular signalling molecules which are often called ligands. The binding of the ligand to a receptor changes the shape of the receptor. These changes are called conformational changes. Conformational changes alone are not enough to cause cellular responses. The ligand is often called the primary messenger because it interacts with the receptor to first send the signal across the cell membrane. We will discuss cell surface receptor dynamics and the binding of ligands to receptors further in Chapter 4. For most of this thesis we will focus on the intracellular part of the signal transduction pathway.

The cell surface receptor and bound ligand act as an external stimulus which triggers events inside the cell. Inside the cell, intracellular signalling pathways are responsible for carrying signals through the interior of the cell to some destination. The destination, which is often the cell nucleus or target proteins, receives the signal and then carries out a cellular response. These responses regulate different cellular functions. The basic parts of a signal transduction pathway are shown in Figure 1.1.

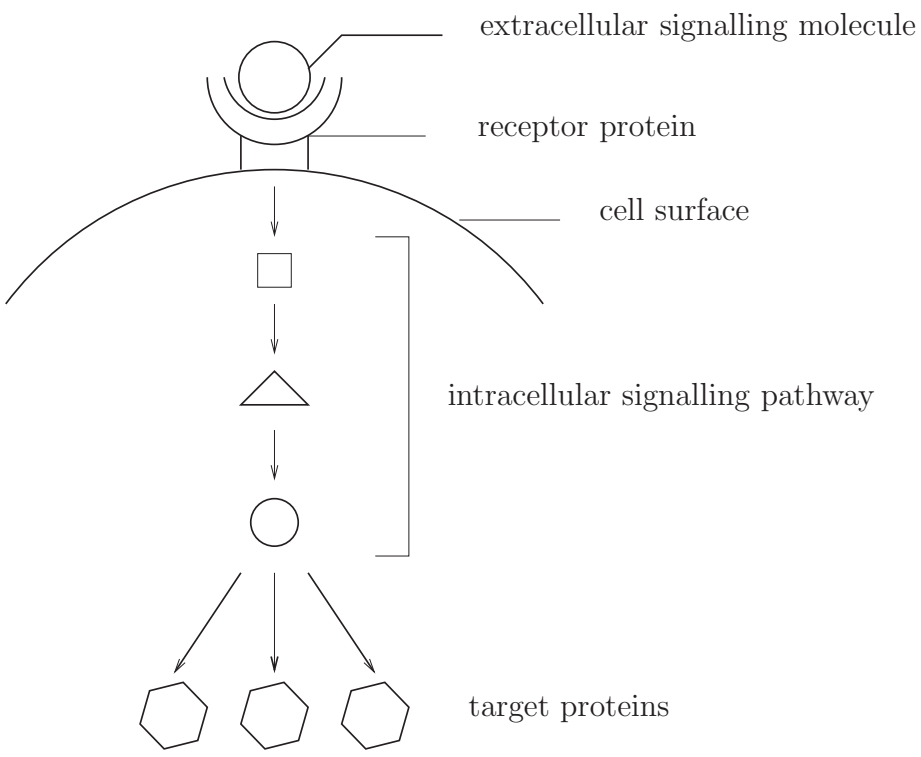


Figure 1.1: A simple signal transduction pathway. The signal transduction pathway is initiated at the cell surface when an extracellular signalling molecule attaches to a cell surface receptor. This acts as a stimulus which activates second messengers within the cell. The different shapes within the intracellular signalling pathway represent different types of signalling proteins and molecules which carry the signal through the interior of the cell. The signal is carried through the interior of the cell through a series of chemical reactions involving enzymes and signalling proteins. The final destination is the cell nucleus or target proteins. When the signal reaches its destination, a cellular decision is made.

Some important components of intracellular signalling pathways are signalling proteins and molecules, and enzymes. Many of these substances are called second messengers. Second messengers carry the signal from the cell surface through the cytoplasm of the cell. Intracellular signalling molecules/proteins can be phosphorylated (activated) or unphosphorylated (deactivated). Kinase enzymes catalyze the reactions which activate signalling proteins and phosphatase enzymes catalyze the reactions which deactivate signalling proteins (see Figure 1.2).

The main part of many intracellular signalling pathways are cascades of protein modification cycles. These cycles propagate external signals from the cell membrane to different targets in the cytoplasm. A protein modification cycle is a system in

which a protein is reversibly modified between two or more states by converter enzymes. For example, a signalling protein can be in an active or inactive state. This protein modification cycle is regulated by the converting kinase and phosphatase enzymes. A protein modification cycle (or cycle for short) is illustrated in Figure 1.2.

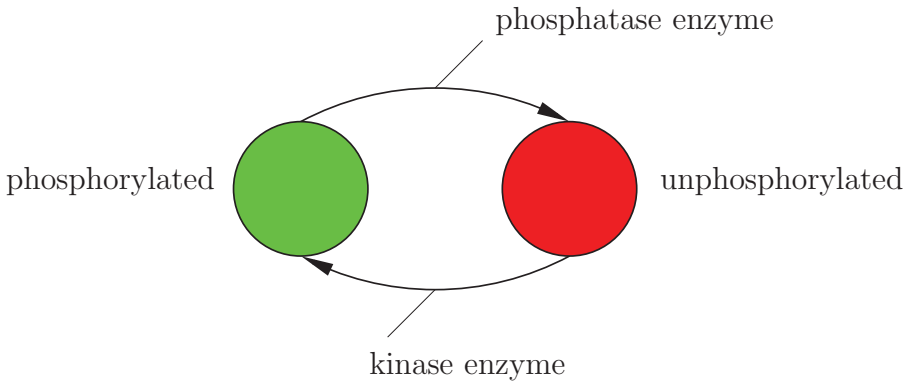


Figure 1.2: A protein modification cycle in which two converter enzymes, a kinase and phosphatase, convert a protein between an active (phosphorylated) and inactive (unphosphorylated) form.

The intracellular portion of a signalling pathway is made of up cycles like those shown in Figure 1.2. Within a cycle, a signalling protein can become phosphorylated and then initiate a similar reaction in the next adjacent protein modification cycle. Another signalling protein becomes phosphorylated in that cycle and then the activation process continues. This process continues through a series of cascading protein modification cycles. These cycles form a cascade–like pathway carrying the signal from the receptor-ligand complex through the interior of the cell to target proteins. The signal can be amplified as it travels down the cascade.

However, the signal is not amplified indefinitely and is eventually terminated. Phosphatase enzymes are one way in which a signalling pathway can be terminated. After the signal is initiated and the information is passed onto its destination, the signalling processes must be terminated. If the signal is not terminated then the cell loses its responsiveness to new signals.

Many signal transduction pathways are mathematically modelled as a series of cascade reactions. A simple pathway without feedback and only feed forward activation is depicted in Figure 1.3. The pathway is made up of protein modification

cycles. At each cycle a reaction converts a signalling protein between its two states, unphosphorylated and phosphorylated. Following stimulation from a cell surface receptor, the protein x is phosphorylated into its active form x_p through a reaction catalyzed by a kinase. Similarly x_p is converted into x through a reaction catalyzed by a phosphatase. The active form of the signalling protein, x_p, then activates the next cycle where y is converted into y_p. This process is continued with y_p activating the next cycle and so on. The signal travels through the cycles in the cascade and results in a cellular response.

Protein modification cycles can also be more complicated. An example of such a cycle is a protein which can be converted between an unphosphorylated form, a phosphorylated form, and a double phosphorylated form. A linear cascade model with forward activation that uses double phosphorylation cycles (or double modification cycles) is shown in Figure 1.4.

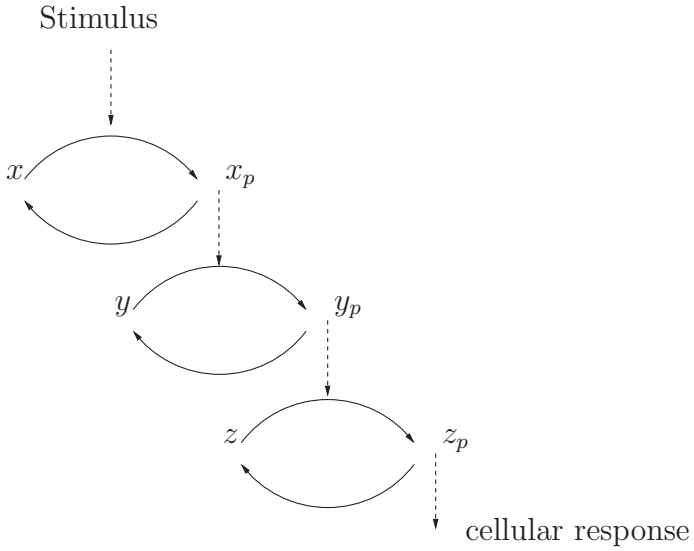


Figure 1.3: A signalling cascade with forward activation and no feedback. The kinase enzymes catalyze the reactions which convert the unphosphorylated forms into the phosphorylated forms. The proteins x, y and z are converted into their phosphorylated forms x_p, y_p, and z_p respectively. The phosphorylated form of each protein activates the next cycle in the cascade. These cascades can be of any length but we have used three cycles for diagram simplicity.

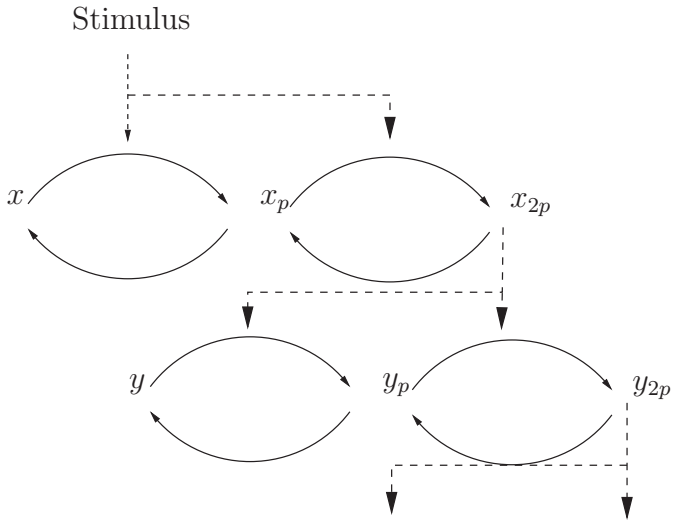


Figure 1.4: A signalling cascade with double phosphorylation cycles. Here the signalling proteins can be in an unphosphorylated form (x), a phosphorylated form (x_p), and a double phosphorylated form (x_{2p}).

We now discuss some mathematical models of signal transduction pathways. Many of these models involve cascades of protein modification cycles. Mathematical models with ordinary differential equations (ODEs) [69, 104, 39, 20, 83] and partial differential equations (PDEs) [9, 73, 77, 13, 93, 51, 94, 30] have been used to study many signal transduction pathways and other cellular signalling phenomenon. These models incorporate important cell signalling behavior such as ultrasensitivity [43, 109], sustained oscillations [47], and bistability (hysteresis) [80, 100, 4].

The majority of signalling models have used ODEs where the cell is viewed as a mixed bag of enzymes and everything in the medium is well mixed [32, 69, 104, 39, 20, 83, 107, 82, 80, 10, 85, 7, 55, 84, 72, 52]. There are hundreds of ODE models of signal transduction. The use of ODEs simplifies the analysis and numerical computations. ODE models have been successful in displaying the complex signalling behavior mentioned above.

We will begin by highlighting some popular ODE models and discuss some of the main cell signalling behaviour observed. One of the first pioneering models using linear cascades, like the one depicted in Figure 1.3, was analyzed in [32]. It is known as the Goldbeter–Koshland (GK) model and is named after the researchers who proposed the model. The GK model consists of a cascade of phosphorylation and dephosphorylation modification cycles. In these cycles a protein is modified between

its phosphorylated and unphosphorylated form. The authors considered pathways with a single cycle as well as pathways with multiple cycles.

We now briefly describe this model. For a single cycle, let the concentration of the phosphorylated form of the protein be denoted as R_p and the concentration of the unphosphorylated form of the protein be denoted as R. The total concentration of the signalling protein is defined as $R_t = R + R_p$ which is constant in time. The reaction can be shown schematically as

$$R \underset{\uparrow A}{\overset{\downarrow S}{\rightleftharpoons}} R_p$$

where the kinase concentration is denoted as S and the phosphatase concentration is denoted as A. Using Michaelis-Menten enzyme kinetics [32], the evolution of R_p is governed by

$$\frac{dR_p}{dt} = \frac{k_1 S(R_t - R_p)}{K_{m1} + R_t - R_p} - \frac{k_2 A R_p}{K_{m2} + R_p}.$$

Note that $R = R_t - R_p$ has been substituted into the ODE. Therefore only one ODE remains for R_p. Here, K_{m1} and K_{m2} are the Michaelis-Menten constants and k_1 and k_2 are maximum rate constants. By studying the above differential equation for a single cycle as well as other more complicated models with multiple cycles, the authors were able to find conditions for ultrasensitivity in the signalling pathway. Ultrasensitivity describes an output response that is more sensitive to stimulus change than the usual Michaelis-Menten response. One of the conditions for ultrasensitivity is that the enzyme concentrations be saturated by the substrates R and R_p. This means that S and A are much smaller than the total concentration R_t. Ultrasensitive behavior is typically represented by a sigmoidal curve as shown in Figure 1.5. The system thus behaves like a switch in response to external signals. The idea of biological switches in signal transduction pathways is important. It is a way in which constant or time dependent stimuli can be converted into an on–off switch.

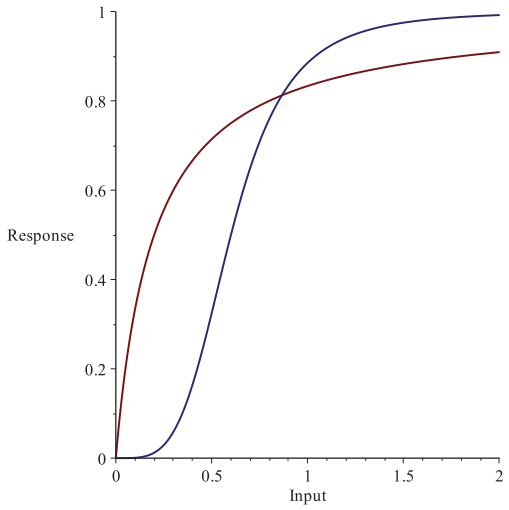


Figure 1.5: Ultrasensitivity is typically represented by a sigmoidal curve (blue). This can be contrasted with a typical Michaelis-Menten response (red).

Similar models to that of [32] by Goldbeter and Koshland have been studied in [59, 33, 34, 68, 108, 11]. These papers focus on the ultrasensitivity of signalling pathways as well as robust switching mechanisms which are a result of the ultrasensitive behaviour possessed by the models.

One specific signal transduction pathway which has received considerable attention is the mitogen-activated protein kinase (MAPK) cascade. MAPK cascades appear in many eukaryotic signal transduction pathways and are found in cells from yeast to mammals [64, 106]. A eukaryotic any organism whose cells contain a nucleus and other organelles enclosed within membranes.

The MAPK cascade is a signalling pathway made up of three levels where an activated kinase at each level phosphorylates the kinase at the next level down through the cascade. The cascade consists of three different types of enzymes. They are MAPK kinase kinase (MAPKKK), a MAPK kinase (MAPKK), and MAPK. MAPKKKs activate MAPKKs by phosphorylation and MAPKKs activate MAPKs by phosphorylation. A MAPK cascade is made up of one single and two double phosphorylation modification cycles [47, 107, 43]. The MAPK cascade has been extensively studied through the use of ODE models [43, 47, 82].

First we will let x_1 correspond to the concentration of MAPKKK-P (phosphorylated MAPK kinase kinase), x_2 correspond to MAPKK-P (phosphorylated MAPK

kinase), $\overline{x}_2$ correspond to MAPKK-PP (double phosphorylated MAPK kinase), x_3 correspond to MAPK-P (phosphorylated MAPK), and $\overline{x}_3$ correspond to MAPK-PP (double phosphorylated MAPK). The dephosphorylated forms are denoted with a tilde. The MAPK cascade is shown schematically in Figure 1.6.

The ODEs for the concentrations of the different enzymes represented in Figure 1.6 have the form

$$\begin{aligned}
\frac{d\widetilde{x}_1}{dt} &= v_2 - v_1, \\
\frac{dx_1}{dt} &= v_1 - v_2, \\
\frac{d\widetilde{x}_2}{dt} &= v_6 - v_3, \\
\frac{dx_2}{dt} &= v_3 - v_6 + v_5 - v_4, \\
\frac{d\overline{x}_2}{dt} &= v_4 - v_5, \\
\frac{d\widetilde{x}_3}{dt} &= v_{10} - v_7, \\
\frac{dx_3}{dt} &= v_7 - v_{10} + v_9 - v_8, \\
\frac{d\overline{x}_3}{dt} &= v_8 - v_9 \,.
\end{aligned} \tag{1.1}$$

The functions v_i model the phosphorylation and dephosphorylation kinetics for the pathway in Figure 1.1. Often Michaelis-Menten kinetics are used. For example, the functions v_1 and v_2 could have the form

$$v_1 = \frac{V_1 \tilde{x}_1}{K_1 + \tilde{x}_1}$$
$$v_2 = \frac{V_2 x_1}{K_2 + x_1},$$

and so on. Here K_1 and K_2 are the Michaelis-Menten constants and V_1 and V_2 are the maximum rate constants. Along with the ODEs, we have the following conservation laws for the total concentrations,

$$\begin{aligned}
x_{1_{\text{total}}} &= \widetilde{x}_1 + x_1, \\
x_{2_{\text{total}}} &= \widetilde{x}_2 + x_2 + \overline{x}_2, \\
x_{3_{\text{total}}} &= \widetilde{x}_3 + x_3 + \overline{x}_3.
\end{aligned}$$

A mathematical model of the MAPK pathway, similar to (1.1), was first developed and studied by Huang and Ferrell in [43]. Based on numerical simulations with

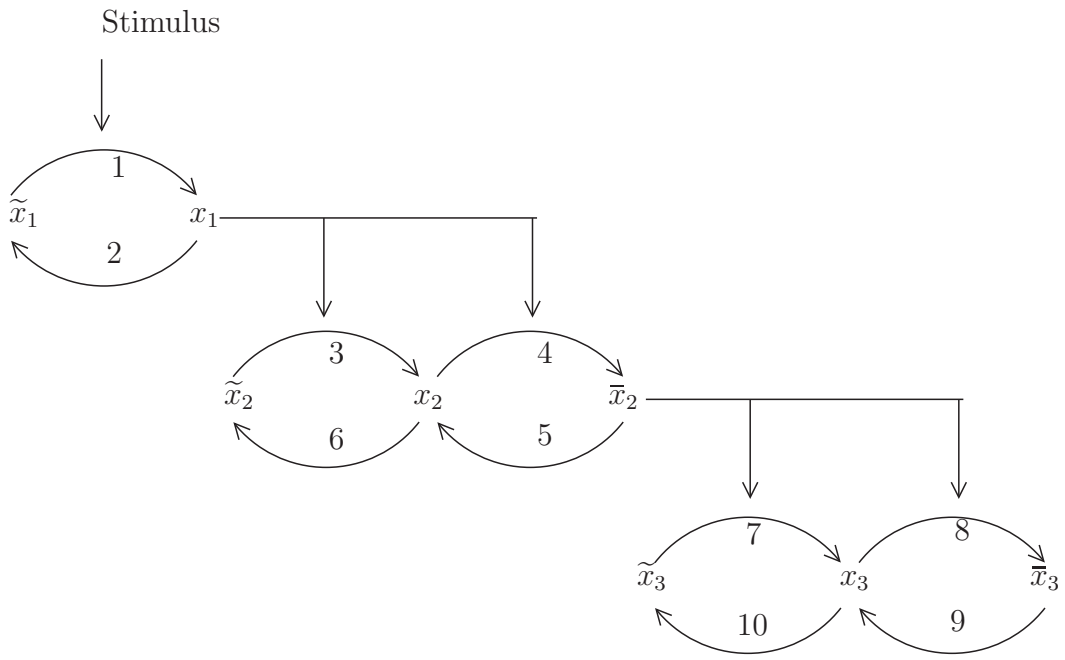


Figure 1.6: The MAPK cascade consists of a single modification cycle as well as two double modification cycles. The labels $i = 1, 2, \ldots, 9$ correspond to the functions v_i appearing in the ODE system (1.1). For example, the function v_1 would be used for the reaction for $\widetilde{x}_1$ into x_1.

hundreds of randomly generated parameter choices, they found that the model could be ultrasensitive. The degree of ultrasensitivity increases as the cascade is descended. This showed that the MAPK cascade can convert different types of stimuli input into a switch like output. This prediction was also verified by biochemical experiments in Xenopus oocyte extracts in the same paper [43]. The stimulus response curves were similar in shape to those given by the Hill equation for $n > 1$. Hill equations have the form

$$y = \frac{x^n}{1 + x^n}.$$

As n increases the shape of this curve becomes sigmoidal and behaves like a switch. Ultrasensitivity is important for the proper functioning of the MAPK cascade because the cascade can then filter out noise, respond more robustly to smaller stimuli, and flip from off to on over a narrow range of stimuli. Switch like behaviour is important in signalling pathways that govern cellular functions such as mitogenesis, cell fate decisions, differentiation, and cell cycle progression.

In later work [27, 28, 110], Ferrell and collaborators showed that ultrasensitivity can lead to bistability in signalling pathways where activated MAPK positively regulates the input to the cascade. Bistability means the system has both a stable off state and a stable on state separated by an unstable state which acts as a threshold. Their results were based on experiments with Xenopus Oocytes as well as some simple quantitative modelling and graphical techniques. They found that a positive feedback loop coupled with a Michaelis-Menten response led to a stable on state and an unstable off state. A positive feedback loop is one in which the output of the signalling pathway first increases the input of the pathway which then also again increases the output. Saying it another way, the input produces more of the output which in turn produces more of the input etc. A negative feedback is the exact opposite. The output produces less of the input which then produces less of the output. Any nonzero level of MAPK activity would trigger the feedback loop and force the system into its on state. On the other hand, they showed that the positive feedback loop coupled with an ultrasensitive response led to bistability. This ensures that the oocyte cannot rest in a state with intermediate MAPK phosphorylation. In summary, the authors found that the MAPK cascades intrinsic ultrasensitivity

coupled with a positive feedback loop resulted in bistability and therefore a robust on-off switch for the pathway.

Switch like behaviour can be either ultrasensitive or an actual true switch between alternate states of a bistable system. It was long thought that positive feedback loops and ultrasensitivity were required in order to have bistability in signalling cascades. Since bistability is responsible for cell differentiation, cell cycle progression, and a type of biochemical memory [60], the question arose as to whether bistability can be generated by mechanisms other than those already found in the literature.

In [72] Kholodenko and co-workers showed that bistability can be generated by double modification cycles, in which the two phosphorylation steps are catalyzed by the same enzyme. However, an analytic study of the conditions and the parameters required for this behaviour was not considered in this paper. In [80], Kholodenko and others analytically demonstrated that double modification of a protein can generate bistability. It should be emphasized that the bistability was found in a single dual phosphorylation–dephosphorylation cycle of MAPK.

Such double modification cycles are illustrated in Figure 1.6. Assuming that the single double modification cycle was the one involving x_2 in Figure 1.6, then the model studied in in [80] consisted of the ODES in (1.1) for $\widetilde{x}_2$, x_2, and $\overline{x}_2$. For the specific Michaelis-Menten enzyme functions used for v_i, refer to [80]. This analysis was done in the absence of a cascade of double modification cycles as well as any imposed feedback regulation. After finding bistability in a single double modification cycle, multistability was then analyzed in cascades of double modification cycles. The necessary kinetic conditions to ensure that bistable behavior was generated were found as well. Multistability occurs in systems that are neither stable nor totally instable. These systems alternate between two or more mutually exclusive states over time.

So far we have highlighted some important papers analyzing ODE models of signal transduction pathways. One of these pathways, the MAPK pathway, has the ability to exhibit ultrasensitive behaviour. This is a characteristic of many signalling cascades in general. We have also pointed out that another intrinsic property of the signalling MAPK cascade is bistability. Double modification cycles alone, coupled with the appropriate enzyme kinetics [80], can lead to bistability in signalling

cascades.

Another important signalling behaviour is sustained oscillations. MAPK cascades often have long feedback loops which can be positive or negative. Whether the loop is positive or negative depends on whether the final kinase in the cascade stimulates or inhibits the activation of the stimulus at the initial level. In [47], Kholodenko showed that a negative feedback loop combined with the ultrasensitive behaviour of the MAPK cascade can lead to sustained oscillations in MAPK phosphorylation. Data was used to predict that the period of oscillations could be anywhere from minutes to hours. A MAPK cascade with a negative feedback loop is shown in Figure 1.7.

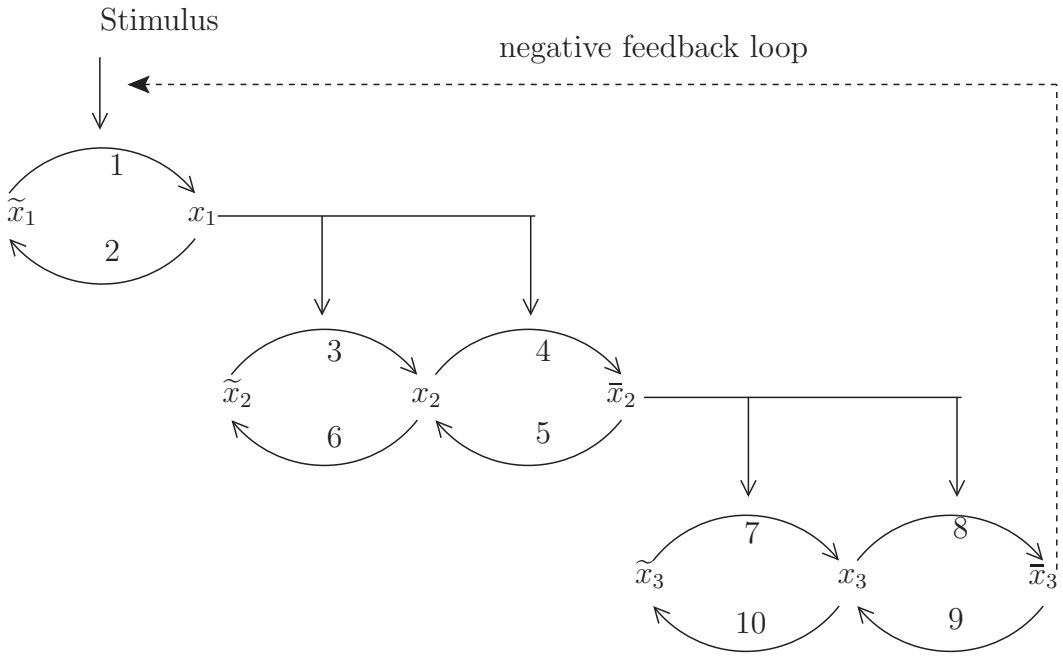


Figure 1.7: The MAPK cascade with a negative feedback loop. The double phosphorylated form of MAPK, with concentration $\overline{x}_3$, inhibits the activation of the stimulus at the initial level.

The model solved in [47], which considers the pathway in Figure 1.7, involves the ODES in (1.1). In the absence of feedback (see Figure 1.6), the function v_1 would have the form

$$v_1 = \frac{V_1 \tilde{x}_1}{K_1 + \tilde{x}_1}.$$

Again, K_1 and K_2 are the Michaelis-Menten constants and V_1 and V_2 are the maximum rate constants. To model the negative feedback as shown in Figure 1.7, Kholodenko chose v_1 to be of the form [47]

$$v_1 = \frac{V_1 \tilde{x}_1}{K_1 + \tilde{x}_1} \cdot \frac{1}{1 + \left(\frac{\bar{x}_3}{K_1}\right)^n}.$$

The exponent n measures the strength of the feedback. The other forms for v_i can be found in [47].

A strong negative feedback can be used to turn off the activation of a cascade. The dynamics in this scenario are associated with a transient response. On the other hand a strong negative feedback may have other effects. In some situations, increasing the level of negative feedback can cause the steady state to lose its stability through a Hopf bifurcation [35]. With this loss of stability, the phosphorylation levels of the cascade kinases begin to oscillate in a sustained way. Kholodenko showed that a negative feedback loop combined with the ultrasensitivity of the MAPK cascade could lead to sustained oscillations in the concentration of the enzymes in the cascade. Later it was shown in [82] that sustained oscillations could be present in an MAPK cascade in the absence of explicit negative feedback. The model considered in that paper was the original Huang-Ferrell model for the MAPK cascade given in [43].

1.2 Spatiotemporal Signalling Dynamics and PDE Models

ODE models have provided numerous insights into the temporal dynamics of signal transduction but there are still many questions about the spatial aspects of signal transduction. Signalling pathways are spatially organized and models with spatiotemporal dynamics can determine how time and space affect signalling pathways and cell fate decisions [50].

To study how the behaviour of a signalling system changes in time and space, PDE models are often used. Reaction diffusion equations are a special class of PDEs and have the form

$$\frac{\partial}{dt}\boldsymbol{U} = \boldsymbol{D}\Delta_{\boldsymbol{x}}\boldsymbol{U} + \boldsymbol{R}(\boldsymbol{U}).$$

Each component of the vector $\boldsymbol{U}(\boldsymbol{x}, t)$ represents the concentration of one substance. The diagonal matrix $\boldsymbol{D}$ contains the diffusion coefficients and $\boldsymbol{R}$ accounts for all the

different reactions. Reaction diffusion equations in signal transduction models can explain the changes in concentrations of signalling proteins and enzymes that are a result of temporal and spatial dynamics.

For example, in many signalling cascades the initiating signals are generated on the cell membrane at cell surface receptors. Then second messengers diffuse through the cytoplasm through a series of cascading reactions. That is, signalling proteins become activated and diffuse to other regions in the cell. The activated proteins then catalyze other reactions among different signalling proteins. These proteins become activated too and then diffuse to other regions in the cell. The components of these pathway are organized in space. ODE models, where everything is assumed to be well mixed, can not account for these spatial features of the signalling pathways or the diffusive transport that is required. Reaction diffusion equations can account for this as well as use geometry which takes into account the importance of cell shape in signal transduction [74, 54, 79].

We now highlight some key concepts which can be incorporated into a cell signalling model consisting of PDEs specified over a spatial domain. The first is the role of scaffolding proteins in signalling cascades. Scaffolding proteins interact with and bind multiple signalling proteins within a signalling cascade, forming them into complexes. Scaffolds can localize the reactions of a signalling pathway to specific regions within the cell (see Figure 1.8). For example, scaffolds can cause phosphorylation of signalling proteins to occur on the surface of compartments within the cell which then further activate downstream pathways [18]. Scaffolds in signalling systems, such as the MAPK pathway, have been reviewed in [58, 91].

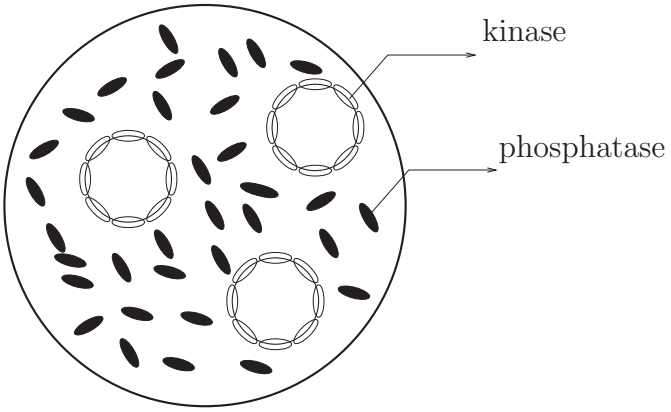


Figure 1.8: In cell signal transduction pathways, activating enzymes (kinase enzymes) and deactivating enzymes (phosphatase enzymes) can separate into different regions and cause spatial gradients of signalling proteins and molecules. Kinase enzymes can localize to subcellular structures while phosphatase enzymes can be spread uniformly throughout the cytosol.

PDE models can incorporate the diffusive transport of signalling proteins and molecules as well as the spatial separation of opposing (activating and deactivating) enzymes within a signalling pathway. Signal transduction pathways make use of cascades of protein modification cycles. The kinase and phosphatase enzymes, which catalyze the activation and deactivation of signalling proteins, can be anchored to specific spatial regions within the cell such as membranes and intracellular compartments (see Figure 1.8). The opposing activator and deactivator enzymes are often spatially separated which can lead to complex signalling gradients within the cell.

For example, signalling proteins can be phosphorylated at the cell membrane due to a membrane bound kinase. The signalling proteins then diffuse through the cytosol away from the membrane where they are deactivated by phosphatase enzymes. Because of this spatial separation of opposing enzymes in space, spatial gradients of phosphorylated signalling proteins occur. Concentrations of the activated proteins are higher near the membrane and lower in the cell interior. The decay rate of these gradients depends on the diffusion rates, decay rates, and the enzyme kinetics.

Signalling spatial gradients of this type have been discovered experimentally in different signalling pathways [45, 111, 29, 76]. Spatial gradients of signalling proteins can coordinate signalling around localized subcellular structures such as scaffolds. Signalling gradients also provide spatial information for key cellular processes such

as cell division. For a thorough discussion on scaffolding proteins, separation and localization of enzymes, and the importance of spatiotemporal dynamics in signal transduction, see the reviews [53, 50, 49].

We will now discuss some PDE models of signal transduction and motivate the PDE model that is studied in the majority of this thesis. The first analysis of intracellular signalling gradients, arising from the spatial separation of opposing enzymes, was under taken by Kholodenko and Brown[13]. This very simple model considers a kinase localized to a spherical subcellular membrane and a phosphatase distributed uniformly throughout the cytoplasm. The authors estimated the relative steady state gradient for a protein that is phosphorylated by the localized kinase and dephosphorylated by the uniformly spread phosphatase. It was assumed that the phosphatase was not saturated by the signalling protein. Along with assuming spherical symmetry, the following 1D PDE for a spherical cell which models diffusion and decay was used,

$$\frac{dp}{dt} = \frac{D}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial p}{\partial r}\right) - k_p p.$$

The diffusion coefficient D is assumed to be constant and the same for the active and inactive forms of the signalling protein. The dephosphorylation rate is k_p. The dependent variable $p(r,t)$ is the concentration of the phosphorylated form of the protein in time and some distance r from the centre of the cell. The steady state solution to this equation is

$$p(r,t) = \frac{C}{r}(e^{\alpha r} - e^{-\alpha r}), \qquad \alpha = \sqrt{k_p/D},$$

where C is a constant of integration. This is assuming there is no diffusive flux at the centre of the cell which implies that $\partial p/\partial r = 0$. The relative difference in the steady state concentration of the phosphorylated form between the cell membrane and the centre of the cell is given by

$$\frac{p(L) - p(0)}{p(0)} = \frac{e^{\alpha L} - e^{-\alpha L}}{2\alpha L} - 1,$$

where L is the radius of the sphere. Therefore, to estimate the relative gradient, only the cell radius, diffusion rate, and dephosphorylation rate are needed. The authors then estimated the potential size of such gradients using experimentally measured

values for these parameters. Similar models were also used by Kholodenko and others [51].

Kholodenko and Stelling [93] studied 1D models using cascades and the spatial separation of enzymes. This 1D geometry corresponds to a cylindrical cell of length H. One of the models in the paper assumed different diffusivities for a signalling protein for its active and inactive forms. It was assumed that a kinase was localized to one pole of the cell (at the coordinate $x = 0$), and the phosphatase was distributed uniformly throughout the cytosol. The kinase activation rate v_a was defined as a boundary condition at $x = 0$ where the activation of the signalling protein occurs. The phosphorylated protein, with concentration $c(x, t)$, diffused through the cell where it was dephosphorylated at a rate v_i. Because the diffusion coefficients for the inactive form, with concentration c_i, and the active form, with concentration c, are different, two PDEs were considered. This is because the total concentration $c_{\text{total}} = c_i(x) + c(x)$ is not constant in space. The model consisted of the following reaction diffusion equations and boundary conditions,

$$\begin{aligned}
\frac{\partial c}{\partial t} &= D\frac{\partial^2 c}{\partial x^2} - v_i(c) \\
\frac{\partial c_i}{\partial t} &= D_i\frac{\partial^2 c_i}{\partial x^2} + v_i(c) \\
-D\frac{\partial c}{\partial x}\Big|_{x=0} = D_i\frac{\partial c_i}{\partial x}\Big|_{x=0} &= v_A; \quad \frac{\partial c}{\partial x}\Big|_{x=H} = \frac{\partial c_i}{\partial x}\Big|_{x=H} = 0.
\end{aligned}$$

Here, v_a corresponds to an activation rate and v_i corresponds to a deactivation rate.

More complicated models involving multiple cycles in a cascade, using extensions of the above model, were also developed. Analysis and simulations were carried out to solve for the steady state solutions. One of the main results was that different diffusivities of the active and inactive forms of the signalling protein could lead to large spatial gradients in the cytoplasm.

Kholodenko and others [73] developed a spatiotemporal model for the MAPK cascade. ODEs were used to describe the evolution of the first single phosphorylated cascade whereas reaction diffusion equations were used to describe the dynamics of the two double modification cycles (see Figure 1.4). The geometry used was a sphere and spherical symmetry was assumed for the diffusion operator. Computer simulations revealed that positive feedback and bistability in the MAPK pathway

could generate a type of travelling wave from the surface deep into the cell interior. This travelling wave of phosphorylated proteins could be a mechanism of carrying a signal over large distances when diffusive transport alone is not enough.

We should say at this point that in this thesis we will not be concerned with how exactly the signal is transferred through the cell. It can happen through the diffusive transport of signalling molecules or it can happen through other mechanisms just mentioned in [73]. In this thesis we will only be considering the changes in concentrations of the signalling proteins. More specifically, we will consider models where the signalling molecules are diffusing through the cell which means they are moving and carrying the signal elsewhere. Therefore, an increase in concentration could be interpreted as an increase in the strength of the signal.

In [9], a 2-state reaction-diffusion system in one dimension (1D) was analyzed and expressions for the steady state concentration profiles were obtained analytically. For related signal transduction models which use PDEs in 1D or use the geometry of a sphere, see [77, 101, 86, 37]. These PDE models and the ones mentioned so far have been in either 1D or used the geometry of a sphere where spherical symmetry is assumed. In some of these papers, the models were simple enough so that the steady state equations could be solved analytically. Others mostly relied on numerical simulations.

Much larger and more complicated models in 2D and 3D have been numerically investigated in [97, 95, 36]. These models are for specific signalling pathways such as the p53–mdm2 oscillatory system, the Hes1 and p53–Mdm2 pathways, and the NF-κB pathways. The finite element package Comsol [1] is used extensively in all these papers to carry out the numerical simulations. Analysis is just not possible on models of this magnitude because of the large number of equations and parameters. The aim of these papers is to investigate numerically the spatiotemporal dynamics of signalling pathways and how factors such as diffusion and cell geometry impact cell signalling dynamics.

Now that we have highlighted some PDE models of signal transduction we will discuss the main works which have motivated this thesis. The first such paper was by Ward and and Straube [94]. In this paper a PDE model with a domain consisting of a sphere with small spherical compartments in its interior was investigated.

This model is relevant because diffusion is in 3D and more importantly the model takes into account the spatial separation and localization of opposing enzymes. The kinase enzymes are localized around the small interior compartments, whereas the phosphatase enzymes are uniformly spread throughout the cytosol. After activation at the compartments, signalling proteins diffuse away from the compartments into the cytosol where they are deactivated. The method of matched asymptotic expansions is used to derive approximate solutions for the steady state concentration profile of the signalling proteins. It was shown that the concentration profiles either decayed algebraically or exponentially, depending on the diffusion rates and dephosphorylation rates. This model consisted of only one type of signalling molecule which could be in an active or deactivated state.

The methods and model from [94] were extended in [61]. In this paper a model of two different signalling proteins, which formed a simple cycle within a cascade, was considered. The system consisted of feed forward activation from the first subcellular compartment to the second as well as negative feedback from the second back to the first. For one specific choice of enzyme kinetic functions, the steady state concentration profiles were found analytically for different parameter scalings.

The model that was used in [61] is the following boundary value problem,

$$\begin{aligned} u_t &= \Delta_x u - \alpha_u^2 u, \quad x \in \Omega_1/\Omega_\varepsilon \\ \partial_{nx} u &= 0, \quad x \in \partial\Omega_1 \\ \varepsilon \partial_{nx} u &= \frac{u}{v+1}, \quad x \in \partial\Omega_{\varepsilon_1} \\ v_t &= \Delta_x v - \alpha_v^2 v, \quad x \in \Omega_1/\Omega_\varepsilon \\ \partial_{nx} v &= 0, \quad x \in \partial\Omega_1 \\ \varepsilon \partial_{nx} v &= u^2, \quad x \in \partial\Omega_{\varepsilon_2}. \end{aligned} \tag{1.2}$$

A more general model of (1.2) is studied in Chapter 2. The PDEs and boundary conditions in (1.2) are defined on a 3D geometry. This geometry consists of a sphere with two small spherical compartments in its domain. The domain is depicted in Figure 1.9.

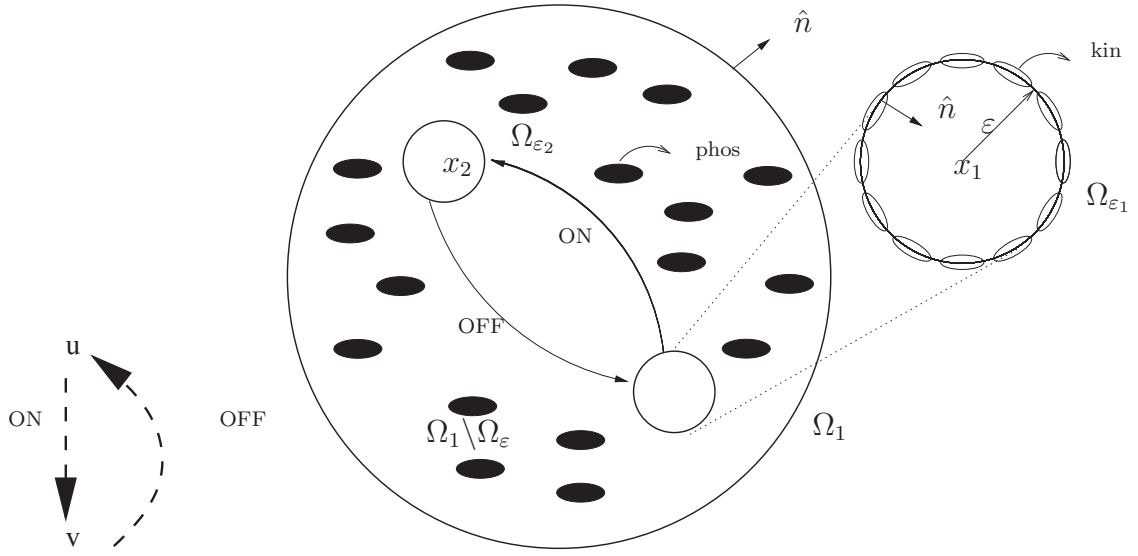


Figure 1.9: Diagram of cell domain with two internal compartments. The normal vector $\hat{n}$ points outward to the cytosolic region, i.e. inward to the interior compartments. The centers of the compartments are the position vectors x_1 and x_2. The activating enzymes (kin) are localized to the surfaces of the two compartments while the deactivating enzymes (phos) are uniformly distributed through the cytosol.

The equations in (1.2) model the interactions between two different types of signalling molecules with concentrations $u(x,t)$ and $v(x,t)$. The signalling molecules with concentration $u(x,t)$ are activated at the first compartment which is centred at x_1. The signalling proteins then diffuse from the first compartment to the second compartment, and activate the second type of signalling molecules with concentration $v(x,t)$. The second type of molecules are activated at the second compartment and then diffuse through the cytosol to the first compartment. Upon reaching the first compartment, the second type of signalling molecules shut off the activation of the first signalling molecules. These dynamics are modelled through the specific choice of enzyme kinetic functions, $u/(v+1)$ and u^2, which appear in the boundary conditions on the subcellular compartments in (1.2). Kinase enzymes are localized to the two subcellular compartments which activate the signalling molecules. During this simple signalling cycle, the signalling molecules are being deactivated throughout the cytosol by phosphatase enzymes which are spread uniformly throughout the domain.

Since the compartments each have a radius of $\varepsilon \ll 1$, asymptotic methods can be used to analyze (1.2). In [61] the method of matched asymptotic expansions was used to find the steady state solutions of (1.2) for different scalings of the decay

parameters α_u and α_v. It was first shown in [94], for a similar model, that if these decay parameters are $O(1)$ then the concentration profiles decay exponentially from the compartments through the cytosol. It was also shown that if these parameters are $O(\sqrt{\varepsilon})$ then the concentration profiles decay algebraically and the solutions are approximately constant in space away from the compartments. This is the same for the model (1.2).

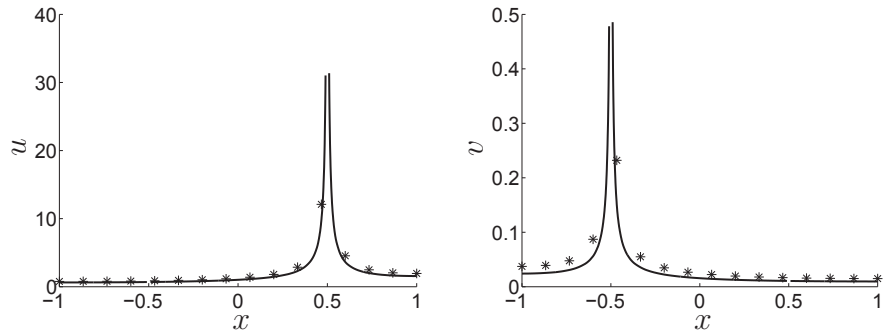
The steady state solutions of (1.2) can be written in terms of two different Green's functions. These functions are the modified Helmholtz Green's function, $G_h(x;x_0)$, and the Neumann Green's function, $G_n(x;x_0)$. These functions, as well as their regular parts, G_R and R_n, will be completely defined in Chapter 2. For now we will write them as,

$$G_h(x;x_0;\alpha) = \frac{e^{-\alpha|x-x_0|}}{4\pi|x-x_0|} + G_R(x;x_0;\alpha)$$
$$G_n(x;x_0) = \frac{1}{4\pi|x-x_0|} + R_n(x;x_0).$$

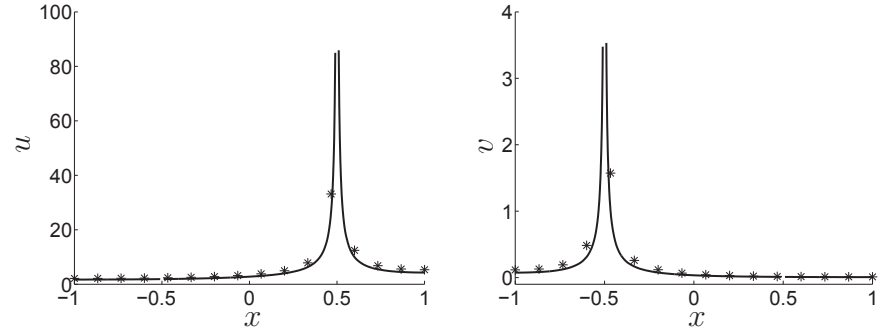
When a decay parameter, α_u or α_v from (1.2), is $O(1)$, the associated steady state is written in terms of the modified Helmholtz Green's function G_h which decays exponentially. When a decay parameter is $O(\sqrt{\varepsilon})$, the associated steady state is written in terms of the Neumann Green's function G_n (with regular part R_N) which decays algebraically. The steady state solutions of (1.2) are summarized in Tables 1.1, 1.2, 1.3 and Figures 1.10, 1.11, 1.12, adapted from [61]. Constants such as c_1, χ_1, χ_2, u_0, and v_0 are defined at the end of the tables.

Table 1.1: Steady state solution of (1.2) with $\alpha_u = O(1)$ and $\alpha_v = O(1)$. The result is plotted in Figure 1.10.

$u(x) = 4\pi c_1 G_h(x;x_1) + O(\varepsilon)$
$v(x) = 64\pi^3 c_1^2 G_h(x_2;x_1)^2 G_h(x;x_2)\varepsilon + O(\varepsilon^2)$
$c_1 = \sqrt{\dfrac{4\pi G_R(x_1;x_1) - \alpha_u}{64\pi^3 G_h(x_1;x_2) G_h(x_2;x_1)^2}}$



(a) $\alpha_u = 1$, $\alpha_v = 1$ (b) $\alpha_u = 1$, $\alpha_v = 1$

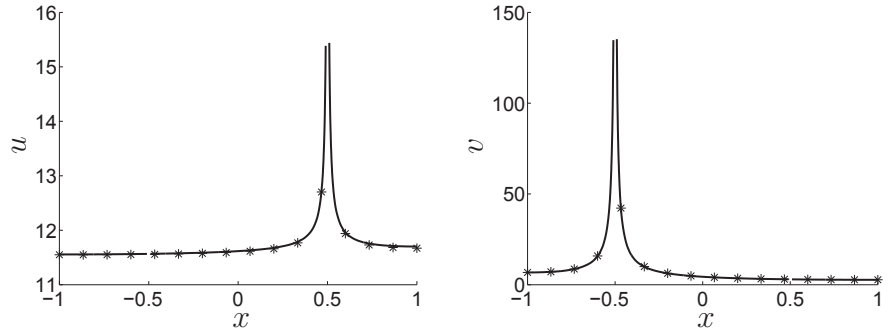


(c) $\alpha_u = 1$, $\alpha_v = 2$ (d) $\alpha_u = 1$, $\alpha_v = 2$

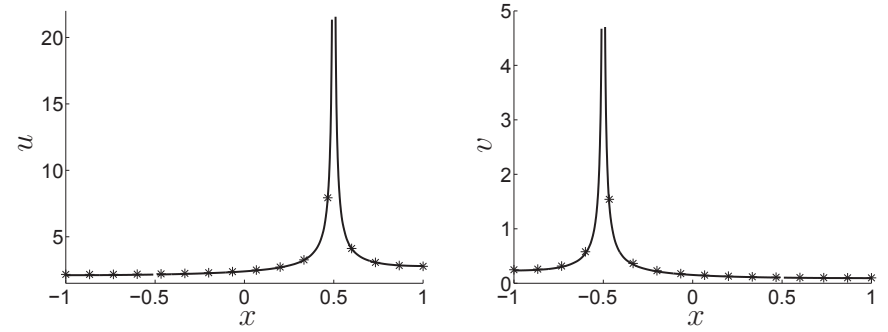
Figure 1.10: Comparison of the asymptotic solutions (star) from Table 1.1 with those from numerical simulations (solid line). Radius of compartments, location of Ω_{ε_1}, and location of Ω_{ε_2} are $\varepsilon = 0.01$, $(0.5, 0, 0)$, and $(-0.5, 0, 0)$, respectively. Here all the solutions exhibit exponential decay.

Table 1.2: Steady state solution of (1.2) with $\alpha_u = \alpha_1\sqrt{\varepsilon}$ and $\alpha_v = O(1)$ where α_1 is $O(1)$. The result is plotted in Figure 1.11.

$u(x) = \frac{1}{\sqrt{\varepsilon}}u_0 + \sqrt{\varepsilon}(4\pi c_1 G_n(x;x_1) + \chi_1) + O(\varepsilon^{3/2})$
$v(x) = 4\pi u_0^2 G_h(x;x_2) + \varepsilon 8\pi u_0(4\pi c_1 G_n(x_2;x_1) + \chi_1)G_h(x;x_2) + O(\varepsilon^2)$
$u_0 = \sqrt{\dfrac{3}{4\pi\alpha_1^2 G_h(x_1;x_2)}} \qquad \chi_1 = \dfrac{R_n(x_1;x_1)}{2u_0 G_h(x_1;x_2)} - \dfrac{4}{3}\pi u_0 \alpha_1^2 G_n(x_2;x_1)$ $c_1 = \dfrac{\alpha_1\sqrt{3}}{6\sqrt{\pi G_h(x_1;x_2)}}$



(a) $\alpha_1 = 1$, $\alpha_v = 1$ (b) $\alpha_1 = 1$, $\alpha_v = 1$

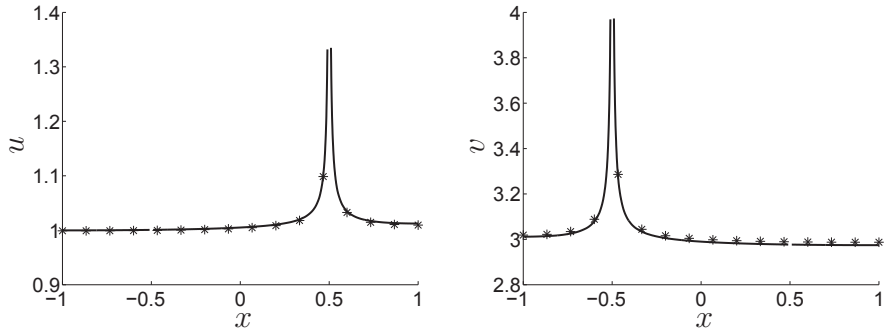


(c) $\alpha_1 = 5$, $\alpha_v = 1$ (d) $\alpha_1 = 5$, $\alpha_v = 1$

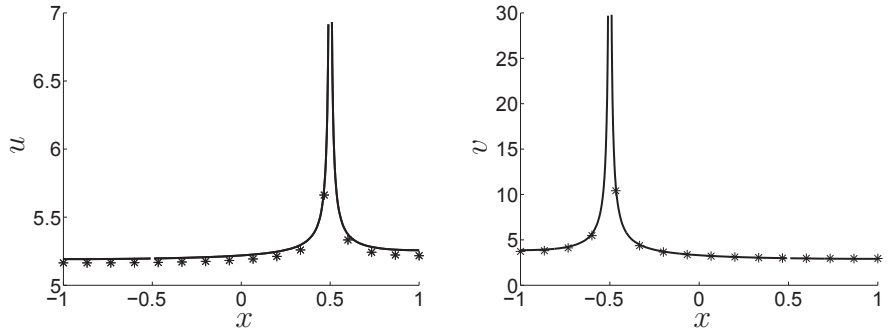
Figure 1.11: Comparison of the asymptotic solutions (star) from Table 1.2 with those from numerical simulations (solid line). The geometry is the same as in Figure 1.10. Here the solution for u decays algebraically and v decays exponentially.

Table 1.3: Steady state solution of (1.2) with $\alpha_u = \alpha_1\sqrt{\varepsilon}$ and $\alpha_v = \alpha_2\sqrt{\varepsilon}$ where α_1 and α_2 are $O(1)$. The result is plotted in Figure 1.12.

$u(x) = u_0 + \varepsilon(4\pi c_1 G_n(x;x_1) + \chi_1) + O(\varepsilon^2)$
$v(x) = v_0 + \varepsilon(4\pi u_0^2 G_n(x;x_2) + \chi_2) + O(\varepsilon^2)$
$v_0 = \frac{3}{\alpha_1^2} \qquad u_0 = \frac{\alpha_2}{\alpha_1} \qquad c_1 = \frac{u_0}{v_0}$ $\chi_2 = 4\pi(R_n(x_1;x_1) - u_0^2 G_n(x_1;x_2))$ $\chi_1 = \frac{\alpha_2^2\chi_2}{6u_0} - 4\pi c_1 G_n(x_2;x_1)$



(a) $\alpha_1 = 1$, $\alpha_2 = 1$ (b) $\alpha_1 = 1$, $\alpha_2 = 1$



(c) $\alpha_1 = 5$, $\alpha_2 = 1$ (d) $\alpha_1 = 5$, $\alpha_2 = 1$

Figure 1.12: Comparison of the asymptotic solutions (star) from Table 1.3 with those from numerical simulations (solid line). The geometry is the same as Figure in 1.10. Here both the solutions for u and v decay algebraically.

If both $\alpha_u = O(\sqrt{\varepsilon})$ and $\alpha_v = O(\sqrt{\varepsilon})$ then the steady state solutions u and v of (1.2) have asymptotic expansions of the form,

$$u(x) = u_0 + \varepsilon u_1(x) + \varepsilon^2 u_2(x) + \dots,$$
$$v(x) = v_0 + \varepsilon v_1(x) + \varepsilon^2 v_2(x) + \dots .$$

where u_0 and v_0 are constants. This can be seen from Table 1.3. Not only do u and v have $O(1)$ expansions for the particular choice of enzyme kinetic functions $u/(v+1)$ and u^2, but these expansions have the same form for any arbitrary functions $F(u, v)$ and $G(u, v)$. This is contrasted with the case when at least one of the decay parameters is $O(1)$. For example, if $\alpha_u = O(1)$ and $\alpha_v = O(1)$ then the expansions for the steady states of u and v in (1.2) have the form (see Table 1.1),

$$u(x) = u_0(x) + \varepsilon u_1(x) + \varepsilon^2 u_2(x) + \dots,$$
$$v(x) = \varepsilon v_1(x) + \varepsilon^2 v_2(x) + \varepsilon^3 v_3(x) + \dots .$$

Moreover, as we will discuss in more detail in Chapter 2, there are also inner expansions which in this case have the form,

$$u^{(i)}(y) = \frac{1}{\varepsilon}u_{-1}^{(i)}(y) + u_0^{(i)}(y) + \varepsilon u_1^{(i)}(y) + \dots,$$
$$v^{(i)}(y) = v_0^{(i)}(y) + \varepsilon v_1^{(i)}(y) + \varepsilon^2 v_2^{(i)}(y) + \dots .$$

In the case that $\alpha_u = O(\sqrt{\varepsilon})$ and $\alpha_v = O(1)$ (see Table 1.2) the steady state solutions u and v have the form,

$$u(x) = \frac{1}{\sqrt{\varepsilon}}u_0 + \sqrt{\varepsilon}u_1(x) + \varepsilon^{3/2}u_2(x) + \dots,$$
$$v(x) = v_0(x) + \varepsilon v_1(x) + \varepsilon^2 v_2(x) + \dots,$$
$$u^{(i)}(y) = \frac{1}{\sqrt{\varepsilon}}u_0^{(i)}(y) + \sqrt{\varepsilon}u_1^{(i)}(y) + \varepsilon^{3/2}u_2^{(i)}(y) + \dots,$$
$$v^{(i)}(y) = \frac{1}{\varepsilon}v_0^{(i)}(y) + v_1^{(i)}(y) + \varepsilon v_2^{(i)}(y) + \dots .$$

The point is, in the case where one of the decay parameters is $O(1)$, the scaling of the asymptotic expansions depends on the particular choice of boundary conditions defined on the compartments.

One of the goals of this thesis is to analyze models of the form (1.2) where the boundary condition functions $u/(v+1)$ and u^2 are replaced by arbitrary functions $F(u,v)$ and $G(u,v)$ respectively. To do the analysis for arbitrary functions F and G, we require that the decay parameters α_u and α_v are both $O(\sqrt{\varepsilon})$. This is because the asymptotic expansions for u and v will all have $O(1)$ expansions whose scaling is independent of the choice of F and G. In Chapter 2 and Chapter 3 it is assumed that the decay parameters α_u and α_v are both $O(\sqrt{\varepsilon})$. In Chapter 4 we drop this assumption and consider only numerical simulations.

When α_u and α_v are both $O(\sqrt{\varepsilon})$ the signalling gradients for u and v will not decay exponentially away from the cellular compartments where activation occurs. It is not difficult for the signal to propagate from one compartment to another or to other regions within the cell because of the weak dephosphorylation rate. This may not be biologically relevant because often these signalling gradients decay exponentially. Often diffusive transport is not enough and there must be other signalling mechanisms which allow the signal to travel over larger distances when the inactivation rate is strong [48]. In this thesis though, we are assuming a weak inactivation rate so that we can carry out the analysis in general.

The models in this thesis are motivated by cell signal transduction pathways in general. We make no attempt though to reproduce experimental quantitative data. Our model is quite small in terms of the number of variables and parameters we use. We usually consider two or three signalling proteins. We could extend our results to larger pathways, such as the MAPK pathway, but that is not the intent of this work. The focus of this thesis is on the mathematical methods and techniques used to analyze such models.

1.3 Models with Delay and Numerical Challenges

An important goal of this thesis will be to consider general models of (1.2) which we do in Chapter 2. Another goal is to study a model which involves adding a time delay to the model studied in Chapter 2. This PDE model with delay will be studied in Chapter 3. The delay is used to model the time lapse during enzyme reactions and also the recovery times associated with conformational changes (changes in the shape of the cell) during the phosphorylation process [92, 40].

It is well known that time delays in differential equations often lead to oscillations. Sustained oscillations are an important feature of some signalling pathways and are often a result of negative feedback loops [47]. Time delay, in addition to negative feedback, is another explanation for observing oscillations in signalling systems. We also note that ODE signalling models which result in sustained oscillations often have many variables and parameters. Using ordinary and partial delay differential equations, sustained oscillations can be observed in much simpler systems.

PDEs with delay are more complicated and less studied than ODEs with delay. The analysis and numerical simulations of delay PDEs in higher dimensions is also challenging. Signalling models incorporating both PDEs and delay are also not well studied. A PDE model in two dimensions with distributed delay of the p53–mdm2 oscillatory system was studied in [36]. In that study, the distributed delay term in the PDE could conveniently be replaced by an additional ODE. Therefore no spatial model with an explicit delay had to be solved numerically. Although there are finite element method (FEM) solvers for PDEs, such as Comsol, they do not have the option to treat problems with delays.

For standard ODEs the derivative of the solution depends only on the solution

at the current time. For delay differential equations (DDEs) the derivative of the solution also depends on the solution at past times. Consider the following example. The ODE

$$\frac{dy}{dt} = ay, \quad y(0) = 1,$$

has the exponential solution

$$y(t) = e^{at}.$$

Now consider the same differential equation but with a constant delay term in the argument of y. This turns the ODE into a DDE,

$$\frac{dy}{dt} = ay(t-T), \quad y(t) = 1 \quad \text{for } t \in [-T, 0]. \tag{1.3}$$

The derivative now depends on the solution in the past. Specifically, it depends on the solution T units back in time. For the initial interval $t \in [0, T]$ we need the values of $y(t)$ for $t \in [-T, 0]$. In contrast to initial value problems (IVPs) for ODEs, IVPs for DDEs require initial data on an interval preceding $t = 0$. We have thus stated in (1.3) that $y(t) = 1$ on the interval $[-T, 0]$. We refer to $y(t)$ on $[-T, 0]$ as the history data. The constant T is called the delay or lag.

The solutions of the above ODE and DDE can be quite different. The above ODE has exponential solutions but if we let $T = 1$ and $a = -\pi/2$ then it is easy to verify by direct substitution that $y(t) = \cos \frac{\pi}{2} t$ is a particular solution of the DDE. The addition of delay often leads to some type of oscillation.

One of the techniques to treat DDEs with constant delay is the method of steps [8]. The method of steps turns a DDE into a series of IVPs for ODEs. It can be used to treat DDEs analytically and numerically. Here we will use the method of steps to solve (1.3) analytically with $a = 1$ and $T = 1$. On the interval $[0, 1]$ the function $y(t-1)$ is accessing history data defined on $[-1, 0]$. Since $y(t) = 1$ for $t \in [-1, 0]$, the DDE on $[0, 1]$ becomes $y'(t) = 1$ with the initial condition $y(0) = 1$. The solution is $y(t) = t + 1$ for $t \in [0, 1]$. We can then solve (1.3) on $[1, 2]$ because we know the required delayed solution on $[0, 1]$. For $t \in [1, 2]$ we have $y'(t) = (t-1) + 1 = t$ along with $y(1) = 2$. The solution is $y(t) = \frac{1}{2}t^2 + \frac{3}{2}$. On the next interval $[2, 3]$ we have $y'(t) = \frac{1}{2}(t-1)^2 + \frac{3}{2}$ and $y(2) = 7/2$. Then $y(t) = \frac{1}{6}(t-1)^3 + \frac{3}{2}t + \frac{1}{3}$ for $t \in [2, 3]$. The solution $y(t)$ as well as its first derivative is plotted in Figure 1.13 for $t \in [-1, 3]$.

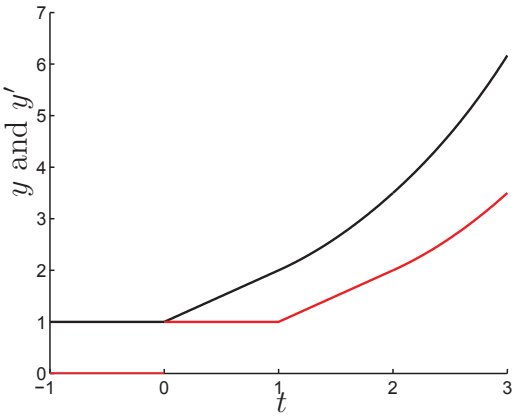


Figure 1.13: The black curve is the solution $y(t)$ and the red curve is the first derivative $y'(t)$. There is a jump discontinuity in the first derivative at $t = 0$ which is propagated as a discontinuity in the second derivative at $t = 1$ and so on. Thus there is a smoothing effect as time evolves forward.

An important remark is that the solution has a jump discontinuity in its first derivative at $t = 0$. Note that $y'(0-) = 0$ and $y'(0+) = 1$. From differentiating (1.3) we have $y''(t) = ay'(t - T)$ and in particular $y''(T) = ay'(0)$. Therefore the discontinuity in the first derivative at $t = 0$ is propagated to a discontinuity in the second derivative at $t = T$. Similarly, the discontinuity in the second derivative at $t = T$ is propagated to a discontinuity in the third derivative at $t = 2T$ and so on. A general property of DDEs is that derivative discontinuities (from now on referred to as simply discontinuities) propagate in time. In contrast, discontinuities can appear in the solution of higher derivatives of an IVP for an ODE, while they are less common.

There are different types of DDEs and each has different properties with regards to discontinuity propagation and smoothing. The most common case in modelling literature is when the delay terms (also called lag functions) are constants [89, 90]. However there is interest in systems of DDEs where the lag functions are time dependent and or state dependent. There are also DDEs which have delayed arguments appearing in the derivative terms. These are called neutral DDEs. They can have very different behavior and pose further theoretical and numerical difficulties. For example, the solutions to neutral DDEs may not get smoother as the integration proceeds [5].

In most cases, numerical methods are used to solve DDEs. The numerical methods used for DDEs are based on the standard methods for ODEs with some additional features. In the appendix of this thesis, in §A.1, we discuss some important ideas which are involved in creating efficient numerical solvers for DDEs. In that section we also mention several specific solvers which are commonly used to solve standard DDEs numerically.

Despite all the advances in numerical solvers for DDEs, solving PDEs with delay is still largely unexplored. Less work has been done on numerical solvers for delay PDEs in 1D and 2D than with numerical solvers for ordinary DDEs. We are not aware of any numerical solvers documented in the literature which can handle delay PDEs in 3D. Also, in higher dimensions, numerical simulations of PDEs with delay are usually carried out using simple geometries such as disks and squares.

Much of the work on delay PDEs focuses on 1D reaction diffusion equations with constant delay [65, 67, 66, 112, 38]. In [44, 102], parabolic PDEs with delay in 1D are considered. In [14] predator prey models in 2D (involving regions that are disks and squares) with delay are studied. In [113], a class of 2D nonlinear delay hyperbolic PDEs is investigated. In the appendix of [113], the authors discuss extending their results to 3D but no simulations are carried out. In all of these papers, numerical schemes for PDEs with delay are developed. A major topic of these papers is on the solvability, stability and convergence of the methods. Numerical simulations are usually carried out at the end of the papers to validate the methods developed.

In Chapter 3 we study a PDE with delay defined over the geometry depicted in Figure 1.9. Because the geometry is complicated, we use the FEM solver Comsol. This software does not allow for delay to be added to the equations. As we will explain in Chapter 3, the delay in our model is only added to the boundary conditions which model the enzyme kinetics on the compartments inside the domain. Since these compartments are small, the solutions are spherically symmetric on and around them. We are able to use Comsol along with the method of steps to solve the PDE model with the delay in the compartment boundary conditions. When storing history data for the solution, we only need to store solution values at singles points within the domain. Then an interpolation function can be used to turn this discrete data into a continuous extension.

Although we can repeat the analysis from Chapter 3 for the case of having the delay in the actual PDE itself, the numerical analysis in this case becomes much more difficult. Now history data needs to be stored and saved over an entire 3D domain. Then an interpolating function in 3D space and time needs to be fitted to that data. We have not considered this case of solving a general 3D PDE with delay over a complicated geometry, and to our knowledge, this has not been considered in the literature.

1.4 Outline of Thesis

An outline of this thesis is as follows. In §2.1 we begin by introducing a generalized model of (1.2). The boundary condition functions $u/(v+1)$ and u^2 are replaced by $F(u,v)$ and $G(u,v)$ respectively. We also assume that $\alpha_u = O(\sqrt{\varepsilon})$ and $\alpha_v = O(\sqrt{\varepsilon})$ for the majority of Chapter 2 (and Chapter 3). Next, in §2.2 we use the method of matched asymptotics to find approximate expressions for the steady state solutions of the model. In §2.3 we carry out a linear stability analysis to derive a linear eigenvalue problem which allows us to determine the stability of the steady state solutions. In §2.4 we introduce a slow time variable which allows us to approximate the system of PDEs with a system of differential algebraic equations (DAEs). This system of DAEs can be converted into a system of ODEs for the examples we consider. The solutions to these systems are used in the asymptotic formulas for the time dependent solution to the PDE model. These solutions are valid when the solution is not changing too quickly. In all the examples considered, we find that the solutions of the DAE systems agree well with the solutions of the PDE model. In §2.5 we apply the results from §2.2 and §2.3 to some specific examples. In these examples we use the results from previous sections to find complex signalling behaviour in the simpler ODE systems such as robust switches, bistability, and sustained oscillations. This behaviour is then confirmed in PDE simulations carried out in Comsol.

In Chapter 3 we repeat the analysis in Chapter 2, but with a time delay added to the model. In §3.1 we rewrite the model from Chapter 2 and then add time delay to get the model which will be analyzed. In §3.2 we derive a nonlinear eigenvalue problem which determines the stability of the equilibrium solutions. In §3.3 we find approximate time dependent solutions of the model. Also in §3.3, the system of PDEs

is approximated by a system of delayed differential algebraic equations (DDAEs). In §3.4 we discuss the numerical method we have implemented, the method of steps combined with Comsol, to solve the 3D PDE model with delay. We discuss a method that we tried to implement which is in its very early stages as well. We also discuss the numerical techniques we have used to solve the DDAEs. In §3.6 we use the Poincaré-Lindstedt method [23] to improve the analysis in the case of a Hopf bifurcation. We also compare numerical results with asymptotic results in §3.5 and §3.6.

In Chapter 4 we introduce a model which includes cell surface receptors. In §4.1 we discuss the dynamics of cell surface receptors and how they tend to cluster together. Then in §4.2 we introduce a model with clusters of receptors on the surface of the cell. These clusters are assumed to be in a stationary state. The geometry used for the model consists of a sphere with N circular patches on its surface. Only numerical simulations are carried out in this chapter with the use of Comsol. We describe how we build this geometry in Comsol too. A flux boundary condition is defined on these patches to model the initiation of the signal transduction pathway. In §4.3 we consider a specific example to examine some of the effects that the number of clusters has on the signalling pathway. In §4.4 we consider an example that brings together the model from Chapter 2 with the cell surface receptors introduced in Chapter 4. In this example the model consists of both cell surface receptors and intracellular compartments with the localized kinase enzymes. Through numerical simulations we show that the number of patches can have a drastic effect on the output of the signalling pathway. Sustained oscillations are sometimes observed depending on the number of clusters.

In Chapter 5 we discuss the results of this thesis and some potential future work. In the appendix we provide all the Matlab and Comsol code we have written to carry out the numerical simulations discussed in this thesis. A short description is provided with each section of code and references it back to the relevant section in the thesis.

Chapter 2

A Model with Localized Enzyme Activation

In this chapter we further extend the results from [94] and [61] by using a similar model and the same geometry. We consider a signalling pathway much like the cascade pathways seen in the ODE models cited earlier. In this chapter full 3D diffusion, enzyme separation and localization, as well as spatiotemporal dynamics are all accounted for. We use the same techniques as in [61], but applied to a more general model. Only one specific choice of enzyme kinetic functions was used in [61], but here we carry out the analysis for any arbitrary choice of functions. We find the steady state solutions analytically and determine their stability which was not considered in [61]. We also approximate the model of PDEs with a system of ODEs which describe the approximate dynamics of the full PDE system. Since the analysis is applied to arbitrary enzyme kinetic functions, we can find conditions for which complex signalling behavior can be observed, such as sustained oscillations, bistability, and robust switching mechanisms.

In §2.1 we begin by describing the model. Next in §2.2 we solve for the steady state solutions for a specific scaling of parameters. Then in §2.3 we do a linear stability analysis to derive an eigenvalue problem which allows us to determine the stability of the steady state solutions. In §2.4 we introduce a slow time variable in to the model which allows us to approximate the system of PDEs with a system of ODEs. Then in §2.5 we apply the analysis from the previous sections to some specific examples.

2.1 Model

The system we derive here can model pathways such as the one seen in Figure 1.3 but with some major differences. We assume that inside the cell there are small subcellular sites where the kinase enzymes are localized. We further assume the phosphatase enzymes are homogenously spread throughout the cytosol. Therefore our model

takes into account the spatial separation of the opposing kinase and phosphatase enzymes. Moreover, our model makes use of scaffolding proteins since all activating reactions caused by the kinase enzymes our localized to subcellular sites. Unlike the ODE models, the signalling proteins in our model will use diffusive transport in 3D to reach other cellular locations. For example, signalling proteins will diffuse to other subcellular compartments and activate the reactions among other signalling proteins. Finally, because the kinase and phosphatase enzymes are separated there will be concentration gradients throughout the cytosol.

In this thesis we are not concerned with the exact mechanisms by which the signal travels through the cell. For example, one way in which the signal travels is by the diffusive transport of the signalling molecules. When the signalling molecules are activated they diffuse to other parts of the cell. This carries the signal to different regions within the cell. The physical movement of individual molecules can carry the signal. There are other ways in which the signal is carried as well. We are only interested in the actual concentrations of the signalling molecules though.

For simplicity we derive the model for a pathway with only two cycles and two signalling proteins. All of the analysis in later sections is done in this context, but it can be easily extended to cascades of any length as seen in the examples in §2.5. We derive a model for proteins which can each be in two different states (active and inactive). Although our model can be applied to cascades where the proteins can be in any number of states.

We choose to model the cell as a sphere of radius R, denoted as Ω_R. Inside the cell there are two spherical compartments, one for each signalling protein. The radii of both compartments are the same and are much smaller than the cell radius. Specifically, the two compartments each have radius εR where $\varepsilon \ll 1$. The kinase enzymes our localized to the surface of the two compartments. The two compartments are denoted as $\Omega_{\varepsilon_1 R}$ and $\Omega_{\varepsilon_2 R}$ and are centred at points $\bar{x}_1$ and $\bar{x}_2$ respectively. We also make the assumption that the distances between the centres of the two compartments as well as to the cell boundary are $O(1)$ compared to the $O(\varepsilon)$ radius of the compartments. We denote the region formed by the union of the two compartments as Ω_ε (see Figure 2.1).

We will use u and v in general to denote the concentrations of the two different

signalling proteins. The total concentration of the first signalling protein is denoted as u_T and the second as v_T. We will denote the concentration of the two states for each as u_a, v_a (phosphorylated/activated) and u_d, v_d (unphosphorylated/deactivated). When the diffusivities are equal for the active and inactive forms then u_T=u_a+u_d and v_T=v_a+v_d, where u_T and v_T are constants provided that u_T and v_T are initially constant in space [49, 93, 51, 94]. Because of the two relationships, u_T=u_a+u_d and v_T=v_a+v_d, it is sufficient to consider the dynamics of the two activated forms, u_a and v_a, alone. If the diffusivities of the phosphorylated and unphosphorylated forms are unequal then u_T and v_T will depend on location. In this case we would have to consider the PDEs governing both the active and inactive forms as considered in [93]. For a given signalling protein, we will assume that the diffusivities of the phosphorylated and unphosphorylated forms are equal.

The activation of the first protein occurs on the surface of $\Omega_{\varepsilon_1 R}$ and the reaction of the second occurs on the surface of $\Omega_{\varepsilon_2 R}$. The kinase enzymes are the enzymes which catalyze these reactions on the compartments within the cell interior. These reactions on the surface of each compartment will be modelled by different Neumann boundary conditions. The opposing reactions which convert deactivate the signalling proteins occur everywhere throughout the cytosol due to the presence of the phosphatases. We also impose a reflecting boundary condition at the surface of the cell. The normal vector is denoted as $\hat{n}$ and it points outward to the cytosolic region. Therefore $\hat{n}$ points inward to the subcellular compartments (see Figure 2.1).

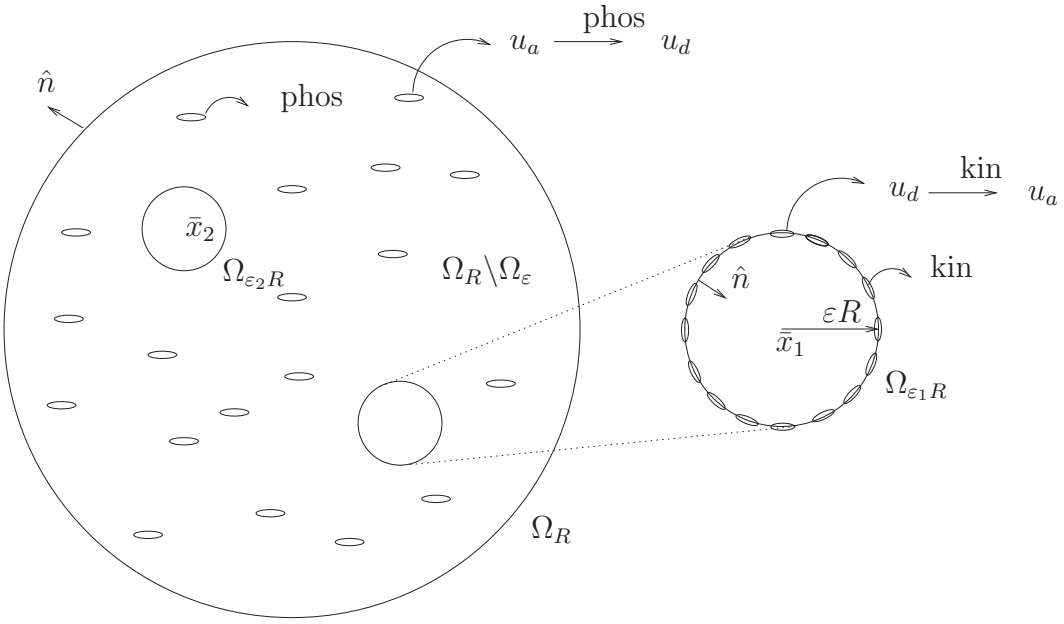


Figure 2.1: The domain is made up of the sphere Ω_R minus the interior of the two smaller compartments $\Omega_{\varepsilon_1 R}$ and $\Omega_{\varepsilon_2 R}$. The kinase (kin) enzymes are localized to the surface of the smaller compartments and the opposing phosphatase (phos) enzymes are spread uniformly throughout the cytosol. The conversion of $u_d \to u_a$ occurs on the boundary of $\Omega_{\varepsilon_1 R}$ and the conversion of $v_d \to v_a$ occurs on the boundary of $\Omega_{\varepsilon_2 R}$. The opposite reactions take place everywhere in the cytosol. The normal vector $\hat{n}$ points outward to the cytosolic region, i.e. inward to the signalling compartments as drawn above. The centres of the compartments are the position vectors $\bar{x}_1$ and $\bar{x}_2$.

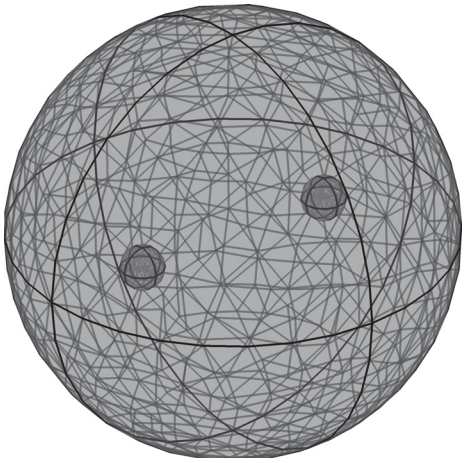


Figure 2.2: Here a typical finite element mesh is represented, which is used in the software package Comsol for the numerical simulations of (2.3). The mesh generator discretizes the domain into tetrahedral, hexahedral, prism, or pyramid mesh elements. The boundaries in the geometry are discretized into triangular or quadrilateral boundary elements. The domain is the unit sphere minus the two small compartments which each have a radius of ε. The interior of the two small compartments is not part of the actual domain for which the PDEs are defined on.

The activated form of the signalling proteins diffuse away from the compartments and the corresponding concentrations decay due to the deactivating phosphatase enzymes. The diffusion coefficients of the two activated proteins, with concentration u_a and v_a, will be denoted as D_u and D_v respectively. We denote the three dimensional Laplace operator as $\Delta_{\bar{x}}$ and $\partial_{n\bar{x}} \equiv \hat{n}\cdot\nabla_{\bar{x}}$ will denote the outward normal derivative to the domain $\Omega_1\backslash\Omega_\varepsilon$. We choose linear decay rates, k_u and k_v, to simplify the analysis.

The evolution for the concentration of the two activated proteins is given by the

following boundary value problem (BVP),

$$\begin{aligned}
\frac{\partial u_a}{\partial t} &= D_u \Delta_{\bar{x}} u_a - k_u u_a, \quad \bar{x} \in \Omega_R \backslash \Omega_{\varepsilon R} \\
\partial_{n\bar{x}} u_a &= 0, \quad \bar{x} \in \partial\Omega_R \\
\varepsilon D_u \partial_{n\bar{x}} u_a &= \frac{R}{3} f(u_d, v_a), \quad \bar{x} \in \partial\Omega_{\varepsilon_1 R}
\end{aligned} \tag{2.1}$$

$$\begin{aligned}
\frac{\partial v_a}{\partial t} &= D_v \Delta_{\bar{x}} v_a - k_v v_a, \quad \bar{x} \in \Omega_R \backslash \Omega_{\varepsilon R} \\
\partial_{n\bar{x}} v_a &= 0, \quad \bar{x} \in \partial\Omega_R \\
\varepsilon D_v \partial_{n\bar{x}} v_a &= \frac{R}{3} g(v_d, u_a), \quad \bar{x} \in \partial\Omega_{\varepsilon_2 R}.
\end{aligned}$$

The two functions f and g model the enzyme kinetics and in this chapter we do the analysis for arbitrary f and g. Here, Ω_ε is defined as the union of the two internal compartments i.e $\Omega_\varepsilon = \Omega_{\varepsilon 1} \cup \Omega_{\varepsilon 2}$. In [94] there is a detailed explanation of why the ε and $R/3$ factors are in the above boundary conditions. In this current derivation we are trying to be consistent with what was done in ([94]). It turns out though that the $R/3$ factor in (2.1) is not so important for the purposes of this thesis. For this thesis we will only be considering models of the form (2.3). The ε scaling appearing in the compartment boundary conditions in (2.1) is important. It results in the flux being strong enough to have an effect globally away from the compartments.

We now introduce the dimensionless variables $u = u_a/u_T$ and $v = v_a/v_T$ into (2.1) as well as $x = \bar{x}/R$. Moreover we define the diffusion times as $\tau_u = R^2/D_u$ and $\tau_v = R^2/D_v$ as well as the parameters $\alpha_u = \sqrt{k_u/D_u}R$ and $\alpha_v = \sqrt{k_v/D_v}R$ which measure the diffusion length in terms of the cell radius. Then (2.1) becomes

$$\begin{aligned}
\tau_u \frac{\partial u}{\partial t} &= \Delta_x u - \alpha_u^2 u, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} u &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon \partial_{nx} u &= \beta_u f(u_T(1-u), v_T v), \quad x \in \partial\Omega_{\varepsilon_1}
\end{aligned} \tag{2.2}$$

$$\begin{aligned}
\tau_v \frac{\partial v}{\partial t} &= \Delta_x v - \alpha_v^2 v, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} v &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon \partial_{nx} v &= \beta_v g(v_T(1-v), u_T u), \quad x \in \partial\Omega_{\varepsilon_2},
\end{aligned}$$

where $\beta_u = \frac{\tau_u}{3u_T}$ and $\beta_v = \frac{\tau_v}{3v_T}$.

Depending on the scaling of α_u^2 and α_v^2 it was shown in [94] that the concentration profiles either decayed exponentially or algebraically, where the mode of decay is described by an associated Green's function. For the majority of this chapter, as well as this thesis, we will consider the case $\alpha_u^2 = O(\varepsilon)$ and $\alpha_v^2 = O(\varepsilon)$. In this case the concentration gradients decay algebraically and therefore this leads to long distance signalling gradients. This corresponds to a stronger signal which can travel between the compartments. As well, when we use the method of matched asymptotic expansions in later sections all the solutions will have $O(1)$ expansions regardless of the flux boundary conditions in (2.2). This allows us to do the analysis for any arbitrary choice of f and g.

Since we assume $\alpha_u^2 = O(\varepsilon)$ and $\alpha_v^2 = O(\varepsilon)$ will write $\alpha_u^2 = \alpha_1^2\varepsilon$ and $\alpha_v^2 = \alpha_2^2\varepsilon$ where α_1 and α_2 are both $O(1)$. For simplicity we will replace the flux boundary conditions in (2.2) with more general functions $F(u,v)$ and $G(u,v)$. Therefore we can rewrite (2.2) as

$$\begin{aligned}
\tau_u \frac{\partial u}{\partial t} &= \Delta_x u - \alpha_1^2 \varepsilon u, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} u &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon \partial_{nx} u &= F(u,v), \quad x \in \partial\Omega_{\varepsilon_1} \\
\\
\tau_v \frac{\partial v}{\partial t} &= \Delta_x v - \alpha_2^2 \varepsilon v, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} v &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon \partial_{nx} v &= G(u,v), \quad x \in \partial\Omega_{\varepsilon_2}.
\end{aligned} \tag{2.3}$$

For the remainder of this thesis we consider models of the form (2.3). We want to study (2.3) both analytically and numerically. We are not concerned with finding solutions of (2.3) and then switching back to (2.2) or (2.1). Therefore, in examples that we consider, we will start with choosing specific functions for F and G. In most of the examples we will ignore the scaling on u and v and also ignore the concentrations of the activated and inactivated forms of the signalling molecules (u_a, u_d, v_a, v_d).

In summary, we will be primarily concerned in finding the steady state solutions of (2.3) and determining their stability, finding time dependent solutions of (2.3),

and finding numerical solutions of (2.3). In Chapter 3 we will introduce a constant time delay into (2.3) and repeat these objectives. With the scaling we have chosen, $\alpha_u^2 = O(\varepsilon)$ and $\alpha_v^2 = O(\varepsilon)$, the dynamics of (2.3) occur over a long time scale. This time scale turns out to be of magnitude $O(1/\varepsilon)$. We could of rescaled time in (2.3) to make this $O(1)$ but we decided to leave it the way it is for notational convenience.

2.2 Steady State Solution

In this section we find asymptotic approximations for the steady state solutions of (2.3). Therefore in this section we are setting $\frac{\partial u}{\partial t} = 0$ and $\frac{\partial v}{\partial t} = 0$ in (2.3). The PDEs in (2.3) are linear but the BVP is complicated by the two holes inside the domain. The key to solving (2.3) is that the holes are of radius $\varepsilon \ll 1$. If we consider (2.3) in the limit as $\varepsilon \to 0$ then we can use methods from asymptotic analysis. In the limit as $\varepsilon \to 0$ the two compartments will reduce to the points at which they are centred, x_1 and x_2. The point x_1 will act as a point source for the solution u and x_2 will act as a point source for v. Near the point x_1, the solution for u will have a singularity and v will have a singularity at x_2. The specific type and strength of the singularities are determined by asymptotically matching an appropriate inner solution together with an outer solution.

When using the method of matched asymptotics there is an outer region and inner region and a corresponding outer and inner solution. Matching is used to match the behavior of the outer and inner solutions. The outer regions are $|x - x_1| \gg \varepsilon$ and $|x - x_2| \gg \varepsilon$ for u and v, respectively. The inner regions in the domain are $|x - x_1| \sim O(\varepsilon)$ and $|x - x_2| \sim O(\varepsilon)$ for u and v, respectively. We first expand the outer and inner solutions in ε where $u^{(i)}$ means the inner solution for u and so on.

$$\begin{aligned} u(x) &= u_0(x) + \varepsilon u_1(x) + \varepsilon^2 u_2(x) + \dots \\ u^{(i)}(y) &= u_0^{(i)}(y) + \varepsilon u_1^{(i)}(y) + \varepsilon^2 u_2^{(i)}(y) + \dots \\ v(x) &= v_0(x) + \varepsilon v_1(x) + \varepsilon^2 v_2(x) + \dots \\ v^{(i)}(y) &= v_0^{(i)}(y) + \varepsilon v_1^{(i)}(y) + \varepsilon^2 v_2^{(i)}(y) + \dots . \end{aligned} \tag{2.4}$$

In (2.4) the inner solutions are written in terms of an inner variable, y, defined as $y = \frac{x - x_j}{\varepsilon}$ where $j = 1$ in the context of u and $j = 2$ in the context of v. The outer problems for u and v come from substituting the outer expansions from (2.4) into

(2.3) and collecting powers in ε which gives

$$0 = \Delta_x u_0, \quad x \in \Omega_1 \backslash \{x_1\}, \qquad \partial_{nx} u_0 = 0, \quad x \in \partial\Omega_1 \tag{2.5a}$$

$$0 = \Delta_x u_1 - \alpha_1^2 u_0, \quad x \in \Omega_1 \backslash \{x_1\}, \qquad \partial_{nx} u_1 = 0, \quad x \in \partial\Omega_1 \tag{2.5b}$$

$$0 = \Delta_x u_2 - \alpha_1^2 u_1, \quad x \in \Omega_1 \backslash \{x_1\}, \qquad \partial_{nx} u_2 = 0, \quad x \in \partial\Omega_1 \tag{2.5c}$$

$$0 = \Delta_x v_0, \quad x \in \Omega_1 \backslash \{x_2\}, \qquad \partial_{nx} v_0 = 0, \quad x \in \partial\Omega_1 \tag{2.5d}$$

$$0 = \Delta_x v_1 - \alpha_2^2 v_0, \quad x \in \Omega_1 \backslash \{x_2\}, \qquad \partial_{nx} v_1 = 0, \quad x \in \partial\Omega_1 \tag{2.5e}$$

$$0 = \Delta_x v_2 - \alpha_2^2 v_1, \quad x \in \Omega_1 \backslash \{x_2\}, \qquad \partial_{nx} v_2 = 0, \quad x \in \partial\Omega_1. \tag{2.5f}$$

For the inner problems for u and v we first use the change of variables $y = (x - x_j)/\varepsilon$ in (2.3). We then substitute the inner expansions from (2.4) into the rescaled version of (2.3) and collect powers of ε again to obtain,

$$0 = \Delta_y u_0^{(i)}, \quad \rho > 1 \tag{2.6a}$$

$$-\partial_\rho u_0^{(i)} = F(u_0^{(i)}, v_0), \quad \rho = 1 \tag{2.6b}$$

$$0 = \Delta_y u_1^{(i)}, \quad \rho > 1 \tag{2.6c}$$

$$-\partial_\rho u_1^{(i)} = F_u(u_0^{(i)}, v_0) u_1^{(i)} + F_v(u_0^{(i)}, v_0) v_1, \quad \rho = 1 \tag{2.6d}$$

$$0 = \Delta_y v_0^{(i)}, \quad \rho > 1 \tag{2.6e}$$

$$-\partial_\rho v_0^{(i)} = G(u_0, v_0^{(i)}), \quad \rho = 1 \tag{2.6f}$$

$$0 = \Delta_y v_1^{(i)}, \quad \rho > 1 \tag{2.6g}$$

$$-\partial_\rho v_1^{(i)} = G_u(u_0, v_0^{(i)}) u_1 + G_v(u_0, v_0^{(i)}) v_1^{(i)}, \quad \rho = 1. \tag{2.6h}$$

In the inner region near the compartments we assume radial symmetry so that $\Delta_y \equiv \partial_\rho^2 + (2/\rho)\partial_\rho$, which is the radially symmetric Laplacian. Here ρ is the radial variable defined as $\rho = |y|$. Note that $\hat{n} \cdot \nabla_y = \partial_{ny} \equiv -\partial_\rho$ because the normal vector, $\hat{n}$, points inward to the spherical compartments (see Figure 2.1).

As already said, in the limit as $\varepsilon \to 0$ the two holes in the domain shrink to the single points x_1 and x_2. These points are excluded from the domain and the behavior of the solution near these points is unknown. The matching condition, known more formally as the Van Dyke matching condition [103], is used to match the outer and inner asymptotic solutions. In the context of u, the matching condition states the behavior of the outer solution $u(x)$ in the limit as $x \to x_1$ must agree with the far-field behavior ($\rho \to \infty$) of the inner solution, $u^{(i)}(y)$, near the hole centred at x_1.

There is a similar matching condition for v. The following matching conditions must hold to all orders in ε,

$$\begin{aligned} u_0 + \varepsilon u_1 + \varepsilon^2 u_2 + \ldots &= u_0^{(i)} + \varepsilon u_1^{(i)} + \varepsilon^2 u_2^{(i)} + \ldots \\ v_0 + \varepsilon v_1 + \varepsilon^2 v_2 + \ldots &= v_0^{(i)} + \varepsilon v_1^{(i)} + \varepsilon^2 v_2^{(i)} + \ldots . \end{aligned} \tag{2.7}$$

From the matching condition in (2.7) we see that $u_0^{(i)} \to u_0$ and $v_0^{(i)} \to v_0$ as $\rho \to \infty$. From (2.5a) and (2.5d) we have that u_0 and v_0 are constants and from (2.6a) and (2.6e), along with their associated boundary conditions in (2.6b) and (2.6f), we have

$$u_0^{(i)} = u_0 + \frac{c_1}{\rho}, \qquad c_1 = F(u_0 + c_1, v_0) \tag{2.8a}$$

$$v_0^{(i)} = v_0 + \frac{c_2}{\rho}, \qquad c_2 = G(u_0, v_0 + c_2). \tag{2.8b}$$

The implicit relationships $c_1 = F(u_0 + c_1, v_0)$ and $c_2 = G(u_0, v_0 + c_2)$ will appear over and over throughout this thesis.

Next we match the outer and inner solutions using (2.7),

$$\begin{aligned} u_0 + \varepsilon u_1(x) + \ldots &= u_0 + \frac{c_1}{|x - x_1|}\varepsilon + \ldots \\ v_0 + \varepsilon v_1(x) + \ldots &= v_0 + \frac{c_2}{|x - x_2|}\varepsilon + \ldots , \end{aligned}$$

which gives

$$\begin{aligned} u_1(x) &\sim \frac{c_1}{|x - x_1|}, \quad x \to x_1 \\ v_1(x) &\sim \frac{c_2}{|x - x_2|}, \quad x \to x_2. \end{aligned} \tag{2.9}$$

From (2.9) we see that both u_1 and v_1 are proportional to the free-space Green's function $G_F(x) = \frac{1}{4\pi|x - x_1|}$ of the Laplace equation, which satisfies $\Delta G_F = -\delta(x - x_1)$. Here $\delta(x)$ is the Dirac delta function. Therefore we add in a δ-term of strength $4\pi c_1$ at the point x_1 and a δ-term of strength $4\pi c_2$ at x_2 which extends the domain of the outer BVPs for u_1 in (2.5b) and v_1 in (2.5e) to all of Ω_1. Therefore u_1 and v_1 are determined by the following BVPs,

$$\Delta_x u_1 - \alpha_1^2 u_0 = -4\pi c_1 \delta(x - x_1), \quad x \in \Omega_1, \qquad \partial_{nx} u_1 = 0, \quad x \in \partial\Omega_1 \tag{2.10a}$$

$$\Delta_x v_1 - \alpha_2^2 v_0 = -4\pi c_2 \delta(x - x_2), \quad x \in \Omega_1, \qquad \partial_{nx} v_1 = 0, \quad x \in \partial\Omega_1. \tag{2.10b}$$

First we integrate (2.10a) and (2.10b) over Ω_1 and apply the Divergence theorem to obtain the following solvability conditions,

$$c_1 = \frac{\alpha_1^2 u_0}{3} \tag{2.11a}$$

$$c_2 = \frac{\alpha_2^2 v_0}{3}. \tag{2.11b}$$

Then we combine (2.8a) with (2.11a) as well as (2.8b) with (2.11b) to obtain the following system of equations which determine the unknown constants u_0 and v_0,

$$\begin{aligned} \frac{\alpha_1^2}{3} u_0 &= F(\delta_1 u_0, v_0) \\ \frac{\alpha_2^2}{3} v_0 &= G(u_0, \delta_2 v_0). \end{aligned} \tag{2.12}$$

Here we have defined

$$\delta_j = 1 + \alpha_j^2/3, \quad j = 1, 2. \tag{2.13}$$

Note that having multiple solutions to (2.12) is not an issue. It just means there are going to be multiple equilibria whose stability is determined in §2.3. If there are complex solutions to (2.12) then they are ignored because they are not physically relevant.

Continuing, we now solve (2.10a) and (2.10b) for u_1 and v_1 to obtain

$$\begin{aligned} u_1(x) &= 4\pi c_1 G_n(x; x_1) + \chi_1 \\ v_1(x) &= 4\pi c_2 G_n(x; x_2) + \chi_2, \end{aligned} \tag{2.14}$$

where χ_1 and χ_2 are constants to to be determined. The solutions in (2.14) are written in terms of the Neumann Green's function $G_n(x; x_0)$, which satisfies

$$\Delta_x G_n(x; x_0) = 1/|\Omega_1| - \delta(x - x_0), \quad x \in \Omega_1 \tag{2.15a}$$

$$\partial_{nx} G_n(x; x_0) = 0, \quad x \in \partial\Omega_1 \tag{2.15b}$$

$$\int_{\Omega_1} G_n(x; x_0)\, dV = 0, \tag{2.15c}$$

where $|\Omega_1| = \frac{4\pi}{3}$, the volume of the unit sphere. For a sphere with radius R we can write $G_n(x; x_0)$ explicitly [6] as

$$G_n(x; x_0) = \frac{1}{4\pi|x - x_0|} + R_n(x; x_0), \tag{2.16}$$

where $R_n(x;x_0)$ is the regular part of this Green's function given by

$$R_n(x;x_0) = \frac{R}{4\pi}\frac{1}{r\left|\frac{R^2}{r^2}x - x_0\right|} + \frac{1}{8\pi R^3}(r^2+r_0^2) - \frac{7}{10\pi R} + \frac{1}{4\pi R}\log\left(\frac{2R^2}{R^2 - rr_0\cos\gamma + r\left|\frac{R^2}{r^2}x - x_0\right|}\right). \tag{2.17}$$

In (2.17) $\cos\gamma = \cos\theta\cos\theta_0 + \sin\theta\sin\theta_0\cos(\phi-\phi_0)$ and (r_0,θ_0,ϕ_0) denotes the spherical coordinates of x_0.

Now we consider the inner problems for $u_1^{(i)}(y)$ and $v_1^{(i)}(y)$ from (2.6c) and (2.6g). We begin by using (2.7) to match and determine the behavior of both $u_1^{(i)}(y)$ and $v_1^{(i)}(y)$ as $\rho\to\infty$,

$$\begin{aligned} u_0 + \varepsilon\left(\frac{c_1}{|x-x_1|} + 4\pi c_1 R_n(x_1;x_1) + \chi_1\right) + \varepsilon^2 u_2 + \ldots &= u_0 + \frac{c_1}{\rho} + \varepsilon u_1^{(i)} \\ v_0 + \varepsilon\left(\frac{c_2}{|x-x_2|} + 4\pi c_2 R_n(x_2;x_2) + \chi_2\right) + \varepsilon^2 v_2 + \ldots &= v_0 + \frac{c_2}{\rho} + \varepsilon v_1^{(i)}. \end{aligned} \tag{2.18}$$

From (2.18) we obtain

$$\begin{aligned} u_1^{(i)} &\to 4\pi c_1 R_n(x_1;x_1) + \chi_1, \quad \rho\to\infty \\ v_1^{(i)} &\to 4\pi c_2 R_n(x_2;x_2) + \chi_2, \quad \rho\to\infty. \end{aligned} \tag{2.19}$$

We then solve (2.6c) and (2.6g) to obtain

$$\begin{aligned} u_1^{(i)} &= 4\pi c_1 R_n(x_1;x_1) + \chi_1 + \frac{A_1}{\rho} \\ v_1^{(i)} &= 4\pi c_2 R_n(x_2;x_2) + \chi_2 + \frac{A_2}{\rho}, \end{aligned} \tag{2.20}$$

where the constants A_1 and A_2 are determined by the associated boundary conditions in (2.6d) and (2.6h),

$$A_1 = F_u(\delta_1 u_0, v_0)(4\pi c_1 R_n(x_1;x_1) + \chi_1 + A_1) + F_v(\delta_1 u_0, v_0)v_1(x_1) \tag{2.21a}$$

$$A_2 = G_u(u_0, \delta_2 v_0)u_1(x_2) + G_v(u_0,\delta_2 v_0)(4\pi c_2 R_n(x_2;x_2) + \chi_2 + A_2). \tag{2.21b}$$

We will use (2.21a) and (2.21b) shortly.

At the next order we have two BVPs for u_2 and v_2 from (2.5c) and (2.5f), respectively. The matching conditions from (2.7) were rewritten in (2.18) and if we substitute $u_1^{(i)}$ and $v_1^{(i)}$ from (2.19) into (2.18) we obtain the following after matching,

$$\begin{aligned} u_2 &\sim \frac{A_1}{|x-x_1|}, \quad x\to x_1 \\ v_2 &\sim \frac{A_2}{|x-x_2|}, \quad x\to x_2. \end{aligned}$$

Similarly to what was done in (2.10), we extend the BVPs in (2.5c) and (2.5f) to all of Ω_1,

$$\Delta_x u_2 - \alpha_1^2 u_1 = -4\pi A_1 \delta(x - x_1), \quad x \in \Omega_1, \qquad \partial_{nx} u_2 = 0, \quad x \in \partial\Omega_1 \tag{2.22a}$$

$$\Delta_x v_2 - \alpha_2^2 v_1 = -4\pi A_2 \delta(x - x_2), \quad x \in \Omega_1, \qquad \partial_{nx} v_2 = 0, \quad x \in \partial\Omega_1. \tag{2.22b}$$

We do not solve (2.22), but if we integrate (2.22a) and (2.22b) over Ω_1 and use the Divergence theorem, and the integral condition from (2.15c) we get

$$A_1 = \frac{\alpha_1^2 \chi_1}{3} \tag{2.23a}$$

$$A_2 = \frac{\alpha_2^2 \chi_2}{3}. \tag{2.23b}$$

Combining (2.21a) with (2.23a) and (2.21b) with (2.23b) we obtain

$$\begin{aligned} \frac{\alpha_1^2}{3}\chi_1 &= F_u(\delta_1 u_0, v_0)(4\pi c_1 R_n(x_1; x_1) + \delta_1 \chi_1) + F_v(\delta_1 u_0, v_0) v_1(x_1) \\ \frac{\alpha_2^2}{3}\chi_2 &= G_u(u_0, \delta_2 v_0) u_1(x_2) + G_v(u_0, \delta_2 v_0)(4\pi c_2 R_n(x_2; x_2) + \delta_2 \chi_2), \end{aligned} \tag{2.24}$$

where we have used the definition of δ_j from (2.13). Note we are using the notation $v(x_1)$and $u(x_2)$ to mean $v_1(x_1) \equiv v_1\big|_{|x-x_1|=\varepsilon}$ and $u_1(x_2) \equiv u_1\big|_{|x-x_2|=\varepsilon}$. We now simplify the system of linear equations in (2.24) for χ_1 and χ_2. In doing so we will write $F_{\_}$ instead of $F_{\_}(\delta_1 u_0, v_0)$ and $G_{\_}$ instead of $G_{\_}(u_0, \delta_2 v_0)$.

Using the results from (2.11), (2.14), as well as (2.16) we can rewrite the system in (2.24) as

$$\mathbf{M} \begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} = \mathbf{M}_1 \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \tag{2.25}$$

where

$$\mathbf{M} = \begin{pmatrix} \frac{\alpha_1^2}{3} - \delta_1 F_u & -F_v \\ -G_u & \frac{\alpha_2^2}{3} - \delta_2 G_v \end{pmatrix} \tag{2.26a}$$

$$\mathbf{M}_1 = \begin{pmatrix} 4\pi F_u R_n(x_1; x_1) & F_v\left(\frac{1}{|x_1 - x_2|} + 4\pi R_n(x_1; x_2)\right) \\ G_u\left(\frac{1}{|x_2 - x_1|} + 4\pi R_n(x_2; x_1)\right) & 4\pi G_v R_n(x_2; x_2) \end{pmatrix}. \tag{2.26b}$$

In all we have obtained the following asymptotic expansions for the solution of the steady state of (2.3),

$$\begin{aligned} u(x) &= u_0 + \varepsilon\left(\frac{c_1}{|x - x_1|} + 4\pi c_1 R_n(x; x_1) + \chi_1\right) + O(\varepsilon^2) \\ v(x) &= v_0 + \varepsilon\left(\frac{c_2}{|x - x_2|} + 4\pi c_2 R_n(x; x_2) + \chi_2\right) + O(\varepsilon^2). \end{aligned} \tag{2.27}$$

The constants u_0 and v_0 are determined from solving (2.12) and χ_1 and χ_2 are determined from solving (2.25). The constants c_1 and c_2 are defined in (2.11).

2.3 Stability Analysis

In §2.2 we found asymptotic approximations to the steady state solution of (2.3) in the limit as $\varepsilon \to 0$. In this section we determine the stability of these steady state approximations. In this section we assume that $\tau_u = O(1)$, $\tau_v = O(1)$, $k_u = O(\varepsilon)$, and $k_v = O(\varepsilon)$. Since $\alpha_u^2 = k_u\tau_u$ and $\alpha_v^2 = k_v\tau_v$ then we again have that $\alpha_u^2 = O(\varepsilon)$ and $\alpha_v^2 = O(\varepsilon)$. We will denote the steady state solutions from (2.27) as $u_E \equiv u(x)$ and $v_E \equiv v(x)$.

We begin the linear stability analysis by introducing small perturbations off of the steady states,

$$\begin{aligned} u &= u_E + \phi(x)e^{\lambda t}, \quad \phi \ll u_E \\ v &= v_E + \psi(x)e^{\lambda t}, \quad \psi \ll v_E. \end{aligned}$$

We substitute these into (2.3) and linearize to obtain

$$\begin{aligned} \tau_u\lambda\phi &= \Delta_x\phi - \alpha_1^2\varepsilon\phi, \quad x \in \Omega_1\backslash\Omega_\varepsilon \\ \partial_{nx}\phi &= 0, \quad x \in \partial\Omega_1 \\ \varepsilon\partial_{nx}\phi &= F_u(u_E, v_E)\phi + F_v(u_E, v_E)\psi, \quad x \in \partial\Omega_{\varepsilon_1} \\ \\ \tau_v\lambda\psi &= \Delta_x\psi - \alpha_2^2\varepsilon\psi, \quad x \in \Omega_1\backslash\Omega_\varepsilon \\ \partial_{nx}\psi &= 0, \quad x \in \partial\Omega_1 \\ \varepsilon\partial_{nx}\psi &= G_u(u_E, v_E)\phi + G_v(u_E, v_E)\psi, \quad x \in \partial\Omega_{\varepsilon_2}, \end{aligned} \tag{2.28}$$

We again use the method of matched asymptotics to analyze (2.28). Here the inner boundary conditions are linear. Just as in §2.2 the outer and inner solutions for ϕ and ψ will be $O(1)$. On the other hand $\lambda = O(\varepsilon)$. The expansions are as follows,

$$\begin{aligned} \phi(x) &= \phi_0(x) + \varepsilon\phi_1(x) + \varepsilon^2\phi_2(x) + \dots \\ \phi^{(i)}(y) &= \phi_0^{(i)}(y) + \varepsilon\phi_1^{(i)}(y) + \varepsilon^2\phi_2^{(i)}(y) + \dots \\ \psi(x) &= \psi_0(x) + \varepsilon\psi_1(x) + \varepsilon^2\psi_2(x) + \dots \\ \psi^{(i)}(y) &= \psi_0^{(i)}(y) + \varepsilon\psi_1^{(i)}(y) + \varepsilon^2\psi_2^{(i)}(y) + \dots \\ \lambda &= \lambda_1\varepsilon + \lambda_2\varepsilon^2 + \dots \end{aligned} \tag{2.29}$$

The form of the asymptotic expansions in (2.29) are chosen to provide a consistent matching between the inner and outer regions. More specifically, integrating the first PDE in (2.28) over the entire domain and applying the divergence theorem yields

$$\int_{\Omega_1 \backslash \Omega_\varepsilon} (\tau_u \lambda \phi + \alpha_1^2 \varepsilon \phi)\, dx = 4\pi\varepsilon (F_u(u_E, v_E)\phi + F_v(u_E, v_E)\psi)\Big|_{|x-x_1|=\varepsilon}$$

For both sides of the above equation to have the same order of magnitude, it must be that $\lambda = O(\varepsilon)$. We could use the same technique with the second PDE in (2.28) two get a second equation. The two equations coupled together give insight into what the orders of the asymptotic expansions should be. For more details refer to the analysis in [94, 61].

Since the following analysis will be similar to that of §2.2 we will not go through as much of the details. The outer problems are

$$\begin{aligned}
&0 = \Delta_x \phi_0, \quad x \in \Omega_1 \backslash \{x_1\}, && \partial_{nx}\phi_0 = 0, x \in \partial\Omega_1 \\
&0 = \Delta_x \phi_1 - (\alpha_1^2 + \tau_u \lambda_1)\phi_0, \quad x \in \Omega_1 \backslash \{x_1\}, && \partial_{nx}\phi_1 = 0, x \in \partial\Omega_1 \\
&0 = \Delta_x \phi_2 - (\alpha_1^2 + \tau_u \lambda_1)\phi_1 - \tau_u \lambda_2 \phi_0, \quad x \in \Omega_1 \backslash \{x_1\}, && \partial_{nx}\phi_2 = 0, x \in \partial\Omega_1 \\
&0 = \Delta_x \psi_0, \quad x \in \Omega_1 \backslash \{x_2\}, && \partial_{nx}\psi_0 = 0, x \in \partial\Omega_1 \\
&0 = \Delta_x \psi_1 - (\alpha_2^2 + \tau_v \lambda_1)\psi_0, \quad x \in \Omega_1 \backslash \{x_2\}, && \partial_{nx}\psi_1 = 0, x \in \partial\Omega_1 \\
&0 = \Delta_x \psi_2 - (\alpha_2^2 + \tau_v \lambda_1)\psi_1 - \tau_v \lambda_2 \psi_0, \quad x \in \Omega_1 \backslash \{x_2\}, && \partial_{nx}\psi_2 = 0, x \in \partial\Omega_1,
\end{aligned}$$

while the inner problems are

$$0 = \Delta_y \phi_0^{(i)}, \quad \rho > 1 \tag{2.30a}$$

$$-\partial_\rho \phi_0^{(i)} = F_u \phi_0^{(i)} + F_v \psi_0, \quad \rho = 1 \tag{2.30b}$$

$$0 = \Delta_y \phi_1^{(i)}, \quad \rho > 1 \tag{2.30c}$$

$$-\partial_\rho \phi_1^{(i)} = F_u \phi_1^{(i)} + F_v \psi_1 + \phi_0^{(i)}(F_{uu} u_1^{(i)} + F_{uv} v_1) + \psi_0 (F_{uv} u_1^{(i)} + F_{vv} v_1), \quad \rho = 1 \tag{2.30d}$$

$$0 = \Delta_y \psi_0^{(i)}, \quad \rho > 1 \tag{2.30e}$$

$$-\partial_\rho \psi_0^{(i)} = G_u \phi_0 + G_v \psi_0^{(i)}, \quad \rho = 1 \tag{2.30f}$$

$$0 = \Delta_y \psi_1^{(i)}, \quad \rho > 1 \tag{2.30g}$$

$$-\partial_\rho \psi_1^{(i)} = G_u \phi_1 + G_v \psi_1^{(i)} + \phi_0 (G_{uu} u_1 + G_{uv} v_1^{(i)}) + \psi_0^{(i)} (G_{uv} u_1 + G_{vv} v_1^{(i)}), \quad \rho = 1. \tag{2.30h}$$

Recall that $F_{\_} = F_{\_}(u_0^{(i)}, v_0)$, $G_{\_} = G_{\_}(u_0, v_0^{(i)})$ and that $u_0^{(i)} = \delta_1 u_0$ and $v_0^{(i)} = \delta_2 v_0$ when $\rho = 1$.

Just as in §2.2 the form of the solutions for $\phi(x)$ and $\psi(x)$ will be

$$\begin{aligned} \phi &= \phi_0 + \varepsilon\phi_1 + \ldots, \quad \phi_1 = \frac{e_1}{|x - x_1|} + 4\pi e_1 R_n(x; x_1) + \chi_3 \\ \psi &= \psi_0 + \varepsilon\psi_1 + \ldots, \quad \psi_1 = \frac{e_2}{|x - x_2|} + 4\pi e_2 R_n(x; x_2) + \chi_4, \end{aligned} \tag{2.31}$$

where ϕ_0 and ψ_0 are constants. We solve (2.30a) and (2.30e) and use their associated boundary conditions in (2.30b) and (2.30f) to get

$$\begin{aligned} e_1 &= F_u(\delta_1 u_0, v_0)(\phi_0 + e_1) + F_v(\delta_1 u_0, v_0)\psi_0 \\ e_2 &= G_u(u_0, \delta_2 v_0)\phi_0 + G_v(u_0, \delta_2 v_0)(\psi_0 + e_2). \end{aligned} \tag{2.32}$$

Then from matching ϕ with $\phi^{(i)}$ and ψ with $\psi^{(i)}$ we can extend the BVP for ϕ_1 and ψ_1,

$$\begin{aligned} \Delta_x\phi_1 - (\alpha_1^2 + \tau_u\lambda_1)\phi_0 &= -4\pi e_1\delta(x - x_1), \quad x \in \Omega_1 \\ \Delta_x\psi_1 - (\alpha_2^2 + \tau_v\lambda_1)\psi_0 &= -4\pi e_2\delta(x - x_2), \quad x \in \Omega_1. \end{aligned} \tag{2.33}$$

Integrating the equations in (2.33) yields the following solvability condition,

$$\begin{aligned} e_1 &= \left(\frac{\alpha_1^2 + \tau_u\lambda_1}{3}\right)\phi_0 \\ e_2 &= \left(\frac{\alpha_2^2 + \tau_v\lambda_1}{3}\right)\psi_0. \end{aligned} \tag{2.34}$$

Combining (2.32) and (2.34) we obtain

$$\begin{pmatrix} \frac{3}{\tau_u(F_u - 1)} & 0 \\ 0 & \frac{3}{\tau_v(G_v - 1)} \end{pmatrix} \begin{pmatrix} \frac{\alpha_1^2}{3} - \delta_1 F_u & -F_v \\ -G_u & \frac{\alpha_2^2}{3} - \delta_2 G_v \end{pmatrix} \begin{pmatrix} \phi_0 \\ \psi_0 \end{pmatrix} = \lambda_1 \begin{pmatrix} \phi_0 \\ \psi_0 \end{pmatrix}.$$

This can be rewritten using the matrix $\mathbf{M}$ defined in (2.26a) as

$$\mathbf{TM} \begin{pmatrix} \phi_0 \\ \psi_0 \end{pmatrix} = \lambda_1 \begin{pmatrix} \phi_0 \\ \psi_0 \end{pmatrix}, \tag{2.35}$$

where

$$\mathbf{T} = \begin{pmatrix} \frac{3}{\tau_u(F_u - 1)} & 0 \\ 0 & \frac{3}{\tau_v(G_v - 1)} \end{pmatrix}.$$

The eigenvalue problem in (2.35) determines the leading order term of the eigenvalue $\lambda = \lambda_1\varepsilon + \lambda_2\varepsilon^2$. It is the eigenvalues of $\mathbf{TM}$ that determine the stability of the equilibrium solutions found in §2.2.

At the next order we need to solve (2.30c) and (2.30g). From these BVP's and matching we obtain

$$\begin{aligned} \phi_1^{(i)} &= 4\pi e_1 R_n(x_1; x_1) + \chi_3 + \frac{A_3}{\rho} \\ \psi_1^{(i)} &= 4\pi e_2 R_n(x_2; x_2) + \chi_4 + \frac{A_4}{\rho}. \end{aligned} \tag{2.36}$$

Before we determine A_3 and A_4 we will first match and extend the domain for the BVPs for ϕ_2 and ψ_2,

$$\begin{aligned} &\Delta_x\phi_2 = (\alpha_1^2 + \tau_u\lambda_1)\phi_1 + \tau_u\lambda_2\phi_0 - 4\pi A_3\delta(x - x_1), \quad x \in \Omega_1, \qquad \partial_{nx}\phi_2 = 0, x \in \partial\Omega_1 \\ &\Delta_x\psi_2 = (\alpha_2^2 + \tau_v\lambda_1)\psi_1 + \tau_v\lambda_2\psi_0 - 4\pi A_4\delta(x - x_2), \quad x \in \Omega_1, \qquad \partial_{nx}\psi_2 = 0, x \in \partial\Omega_1. \end{aligned}$$

Integrating the above BVP, applying the Divergence theorem, and the integral condition in (2.15c) gives

$$\begin{aligned} A_3 &= \frac{(\alpha_1^2 + \tau_u\lambda_1)\chi_3 + \tau_u\lambda_2\phi_0}{3} \\ A_4 &= \frac{(\alpha_2^2 + \tau_v\lambda_1)\chi_4 + \tau_v\lambda_2\psi_0}{3}. \end{aligned} \tag{2.37}$$

The boundary conditions to determine A_3 and A_4 are (2.30d) and (2.30h). Using these as well as (2.31), (2.36), and (2.37) we obtain after simplifying

$$\begin{aligned} (\mathbf{TM} - \lambda_1\mathbf{I}) \begin{pmatrix} \chi_3 \\ \chi_4 \end{pmatrix} - \lambda_2 \begin{pmatrix} \phi_0 \\ \psi_0 \end{pmatrix} = \mathbf{TM}_1 \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} + \\ \mathbf{T} \begin{pmatrix} \delta_1 + \frac{\tau_u\lambda_1}{3} & 0 \\ 0 & \delta_2 + \frac{\tau_v\lambda_1}{3} \end{pmatrix} \begin{pmatrix} \phi_0 & 0 \\ 0 & \psi_0 \end{pmatrix} \left(\mathbf{M}_2 \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} + \mathbf{M}_{2\delta} \begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} \right) \\ + \mathbf{T} \begin{pmatrix} \psi_0 & 0 \\ 0 & \phi_0 \end{pmatrix} \left(\mathbf{M}_3 \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} + \mathbf{M}_{3\delta} \begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} \right). \end{aligned} \tag{2.38}$$

The matrices that have not been defined previously are

$$\mathbf{M}_2 = \begin{pmatrix} 4\pi R_n(x_1;x_1)F_{uu} & F_{uv}\left(\frac{1}{|x_1-x_2|}+4\pi R_n(x_1;x_2)\right) \\ G_{uv}\left(\frac{1}{|x_2-x_1|}+4\pi R_n(x_2;x_1)\right) & 4\pi R_n(x_2;x_2)G_{vv} \end{pmatrix}$$

$$\mathbf{M}_{2\delta} = \begin{pmatrix} \delta_1 F_{uu} & F_{uv} \\ G_{uv} & \delta_2 G_{vv} \end{pmatrix}$$

$$\mathbf{M}_3 = \begin{pmatrix} 4\pi R_n(x_1;x_1)F_{uv} & F_{vv}\left(\frac{1}{|x_1-x_2|}+4\pi R_n(x_1;x_2)\right) \\ G_{uu}\left(\frac{1}{|x_2-x_1|}+4\pi R_n(x_2;x_1)\right) & 4\pi R_n(x_2;x_2)G_{uv} \end{pmatrix}$$

$$\mathbf{M}_{3\delta} = \begin{pmatrix} \delta_1 F_{uv} & F_{vv} \\ G_{uu} & \delta_2 G_{uv} \end{pmatrix}.$$

For simplicity we rewrite (2.38) as

$$(\mathbf{TM} - \lambda_1 \mathbf{I}) \begin{pmatrix} \chi_3 \\ \chi_4 \end{pmatrix} = \mathbf{N} + \lambda_2 \begin{pmatrix} \phi_0 \\ \psi_0 \end{pmatrix}, \tag{2.39}$$

where $\mathbf{N}$ is the right hand side of (2.38).

A consequence of the Fredholm alternative is that (2.39) has a solution if and only if $\mathbf{N} + \lambda_2 \begin{pmatrix} \phi_0 \\ \psi_0 \end{pmatrix}$ is orthogonal to $\ker((\mathbf{TM} - \lambda_1 \mathbf{I})^*)$. Therefore we can solve

$$0 = \left\langle \mathbf{N} + \lambda_2 \begin{pmatrix} \phi_0 \\ \psi_0 \end{pmatrix}, \mathbf{Y} \right\rangle,$$

for λ_2 where the vector $\mathbf{Y}$ satisfies $(\mathbf{TM} - \lambda_1 \mathbf{I})^* \mathbf{Y} = 0$. Here $\mathbf{A}^*$ denotes the Hermitian/conjugate transpose of $\mathbf{A}$ and $\langle .\ ,\ . \rangle$ is the standard complex inner product. Although calculating λ_2 makes the approximation to the eigenvalue more accurate, for the majority of this chapter we will consider the more simple matrix $\mathbf{TM}$ which gives the leading order term of the eigenvalue. The eigenvalue problem in (2.35) is the main result of this section.

2.4 Time Dependent Approximations of the PDE Model

Thus far we have found asymptotic approximations for the steady state solutions of (2.3) and derived a linear matrix eigenvalue problem do determine their stability. In this section we find an approximate time dependent solution to (2.3) which is

valid for times when the solution is not changing too quickly or when the solution is near equilibrium. The solutions will have the same form as in (2.27) except that the constants in (2.27), which were determined by systems of algebraic equations, will now be functions of time and will be determined by systems of ODEs. Actually, these systems are differential algebraic equations (DAEs) but they can be transformed into ODEs if the algebraic equations can be solved.

First we introduce the rescaled slow time variable $\tau = \varepsilon t$ into (2.3) to obtain

$$\begin{aligned}
\varepsilon\tau_u \frac{\partial u}{\partial \tau} &= \Delta_x u - \alpha_1^2 \varepsilon u, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} u &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon\partial_{nx} u &= F(u,v), \quad x \in \partial\Omega_{\varepsilon_1}
\end{aligned}$$

$$\begin{aligned}
\varepsilon\tau_v \frac{\partial v}{\partial \tau} &= \Delta_x v - \alpha_2^2 \varepsilon v, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} v &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon\partial_{nx} v &= G(u,v), \quad x \in \partial\Omega_{\varepsilon_2}.
\end{aligned}$$

We will find an asymptotic solution to the time dependent problem in (2.40) using the method of matched asymptotic expansions.

We expand the outer and inner solutions in terms of ε

$$\begin{aligned}
u(\tau,x) &= u_0(x,\tau) + \varepsilon u_1(x,\tau) + \dots \\
v(\tau,x) &= v_0(x,\tau) + \varepsilon v_1(x,\tau) + \dots \\
u^{(i)}(\tau,y) &= u_0^{(i)}(y,\tau) + \varepsilon u_1^{(i)}(y,\tau) + \dots \\
v^{(i)}(\tau,y) &= v_0^{(i)}(y,\tau) + \varepsilon v_1^{(i)}(y,\tau) + \dots
\end{aligned}$$

and substitute into (2.40) and equate powers of ε to obtain the following outer

problems

$$\Delta_x u_0 = 0, \quad x \in \Omega_1 \backslash \{x_1\}, \qquad \partial_{nx} u_0 = 0, \quad x \in \partial\Omega_1 \tag{2.40}$$

$$\tau_u \frac{\partial u_0}{\partial \tau} = \Delta_x u_1 - \alpha_1^2 \, u_0, \quad x \in \Omega_1 \backslash \{x_1\}, \qquad \partial_{nx} u_1 = 0, \quad x \in \partial\Omega_1 \tag{2.41}$$

$$\tau_u \frac{\partial u_1}{\partial \tau} = \Delta_x u_2 - \alpha_1^2 u_1, \quad x \in \Omega_1 \backslash \{x_1\}, \qquad \partial_{nx} u_2 = 0, \quad x \in \partial\Omega_1 \tag{2.42}$$

$$\Delta_x v_0 = 0, \quad x \in \Omega_1 \backslash \{x_2\}, \qquad \partial_{nx} v_0 = 0, \quad x \in \partial\Omega_1 \tag{2.43}$$

$$\tau_v \frac{\partial v_0}{\partial \tau} = \Delta_x v_1 - \alpha_2^2 \, v_0, \quad x \in \Omega_1 \backslash \{x_2\}, \qquad \partial_{nx} v_1 = 0, \quad x \in \partial\Omega_1 \tag{2.44}$$

$$\tau_v \frac{\partial v_1}{\partial \tau} = \Delta_x v_2 - \alpha_2^2 v_1, \quad x \in \Omega_1 \backslash \{x_2\}, \qquad \partial_{nx} v_2 = 0, \quad x \in \partial\Omega_1. \tag{2.45}$$

The inner problems appear the exact same as those from (2.6) except that the inner and outer solutions are functions of τ as well.

From (2.40) and (2.43) we have u_0 and v_0 are functions of τ and not space. For the leading order inner solutions we have

$$u_0^{(i)}(y,\tau) = u_0(\tau) + \frac{c_1(\tau)}{\rho}, \quad c_1 = F(u_0 + c_1, v_0)$$
$$v_0^{(i)}(y,\tau) = v_0(\tau) + \frac{c_2(\tau)}{\rho}, \quad c_2 = G(u_0, v_0 + c_2) \quad .$$

Then after matching we get that u_1 and v_1 satisfy

$$\begin{aligned} \tau_u \frac{\partial u_0}{\partial \tau} &= \Delta_x u_1 - \alpha_1^2 u_0 + 4\pi c_1 \delta(x - x_1), \quad x \in \Omega_1, \qquad \partial_{nx} u_1 = 0, \quad x \in \partial\Omega_1 \\ \tau_v \frac{\partial v_0}{\partial \tau} &= \Delta_x v_1 - \alpha_2^2 v_0 + 4\pi c_2 \delta(x - x_2), \quad x \in \Omega_1, \qquad \partial_{nx} v_1 = 0, \quad x \in \partial\Omega_1. \end{aligned} \tag{2.46}$$

Integrating the above leads to the following solvability condition for $u_0(\tau)$ and $v_0(\tau)$,

$$\begin{aligned} \tau_u \frac{\partial u_0}{\partial \tau} &= -\alpha_1^2 u_0 + 3c_1, \quad c_1 = F(u_0 + c_1, v_0) \\ \tau_v \frac{\partial v_0}{\partial \tau} &= -\alpha_2^2 v_0 + 3c_2, \quad c_2 = G(u_0, v_0 + c_2). \end{aligned} \tag{2.47}$$

The system defined in (2.47) is system of DAEs. If we solve the algebraic equations for c_1 and c_2 in terms of u_0 and v_0 then we get a system of two ODEs for u_0 and v_0. One important remark is that there may be multiple solutions for $c_1(u_0(\tau), v_0(\tau))$ and $c_2(u_0(\tau), v_0(\tau))$ from solving the algebraic equations defined in (2.47). For each F and G we must carefully consider all the solution branches for both c_1 and c_2 depending

on the nonlinearity of F and G. In the cases we have studied with multiple solution branches, there is always a unique pair of branches (one for c_1 and one for c_2) which are defined in the region of interest and give the expected dynamical behavior. That is, after selecting the unique pair of branches for c_1 and c_2, the resulting dynamics of the ODEs in (2.47) agree with the analysis from §2.2 and §2.3. In §6.2 we provide the detailed analysis including the selection of solution branches for a specific choice of F and G.

We can write the solution in (2.46) using the Green's function defined in (2.15),

$$\begin{aligned} u_1(x,\tau) &= 4\pi c_1(\tau) G_n(x;x_1) + \chi_1(\tau) \\ v_1(x,\tau) &= 4\pi c_2(\tau) G_n(x;x_2) + \chi_2(\tau), \end{aligned}$$

where $\chi_1(\tau)$ and $\chi_2(\tau)$ are solved through a solvability condition at the next order.

At the next order in the inner solution, $u_1^{(i)}(y,\tau)$ and $v_1^{(i)}(y,\tau)$ will have the exact same form as in (2.20). The functions $A_1(\tau)$ and $A_2(\tau)$ are found from boundary conditions which are identical to the ones in (2.6d) and (2.6h),

$$\begin{aligned} A_1 &= F_u(u_0+c_1,v_0)(4\pi c_1 R_n(x_1;x_1) + \chi_1 + A_1) + F_v(u_0+c_1,v_0)v_1(x_1,\tau) \\ A_2 &= G_u(u_0,v_0+c_2)u_1(x_2,\tau) + G_v(u_0,v_0+c_2)(4\pi c_2 R_n(x_2;x_2) + \chi_2 + A_2). \end{aligned} \tag{2.48}$$

Next, we match and extend the BVPs in (2.42) and (2.45),

$$\begin{aligned} \tau_u \frac{\partial u_1}{\partial \tau} &= \Delta_x u_2 - \alpha_1^2 u_1 + 4\pi A_1 \delta(x-x_1), \quad x \in \Omega_1 \backslash \{x_1\}, \quad \partial_{nx} u_2 = 0, \quad x \in \partial\Omega_1 \\ \tau_v \frac{\partial v_1}{\partial \tau} &= \Delta_x v_2 - \alpha_2^2 v_1 + 4\pi A_2 \delta(x-x_2), \quad x \in \Omega_1 \backslash \{x_2\}, \quad \partial_{nx} v_2 = 0, \quad x \in \partial\Omega_1. \end{aligned}$$

Integrating the above we get the following solvability condition for $\chi_1(\tau)$ and $\chi_2(\tau)$,

$$\begin{aligned} \tau_u \frac{\partial \chi_1}{\partial \tau} &= -\alpha_1^2 \chi_1 + 3A_1 \\ \tau_v \frac{\partial \chi_2}{\partial \tau} &= -\alpha_2^2 \chi_2 + 3A_2. \end{aligned} \tag{2.49}$$

The algebraic equations from (2.48) taken together with the differential equations defined in (2.49) lead to a system of DAEs. Since the equations for A_1 and A_2 are linear we can solve them. Combining (2.48) and (2.49) leads to the following system of ODEs to determine $\chi_1(\tau)$ and $\chi_2(\tau)$,

$$\frac{\partial}{\partial \tau}\begin{pmatrix}\chi_1 \\ \chi_2\end{pmatrix} = \mathbf{TM}\begin{pmatrix}\chi_1 \\ \chi_2\end{pmatrix} - \mathbf{TM_1}\begin{pmatrix}c_1 \\ c_2\end{pmatrix}. \tag{2.50}$$

The main results from this section are the following. In the outer region away from the compartments, the solutions $u(x,\tau)$ and $v(x,\tau)$ are constants in space, $u_0(\tau)$ and $v_0(\tau)$ respectively. Their evolution, as well as $c_1(\tau)$ and $c_2(\tau)$, is governed by the system of DAEs in (2.47). Spatial terms do not show up in this leading order system. At the next order there is a system of ODEs for $\chi_1(\tau)$ and $\chi_2(\tau)$ in (2.50) and this is where the spatial terms appear. Solving both (2.47) and (2.50) allows the solutions for $u(x,\tau)$ and $v(x,\tau)$ to be written as

$$\begin{aligned} u(x,\tau) &= u_0(\tau) + \varepsilon\left(\frac{c_1(\tau)}{|x-x_1|} + 4\pi c_1(\tau)R_n(x;x_1) + \chi_1(\tau)\right) \\ v(x,\tau) &= v_0(\tau) + \varepsilon\left(\frac{c_2(\tau)}{|x-x_2|} + 4\pi c_2(\tau)R_n(x;x_2) + \chi_2(\tau)\right). \end{aligned} \tag{2.51}$$

For most choices of F and G it will not be possible to solve the ODEs in (2.47) and (2.50) analytically. They can be solved numerically though, and much faster than the PDE system in (2.3). In §2.5 we compare the numerical solutions from the ODE systems in this section with the full 3D numerics from (2.3) and there is very good agreement.

2.4.1 ODE system for different scaling of α_v

This is the only part of the thesis were we take a small detour and consider a different scaling for the decay rates in (2.3). In this section we consider the BVP in (2.3) but instead of assuming both $\alpha_u^2 = O(\varepsilon)$ and $\alpha_v^2 = O(\varepsilon)$ we will now let $\alpha_v^2 = O(1)$. This problem was studied in [61] and the steady state solution was found for a specific choice of $F(u,v)$ and $G(u,v)$ which we again use here. We want to show that the same analysis in §2.4 when done with this different scaling of α_v leads to something different than the leading order system of DAEs in (2.47). More specifically, the distance between the compartments comes into the leading order ODE where as it did not in (2.47). With $\tau_u = \tau_v = 1$, $k_u = k_1\varepsilon$, $k_v = k_2$, where $k_1 = O(1)$ and $k_2 = O(1)$, and the particular choice of $F = u/(v+1)$ and $G = u^2$ then (2.3) becomes

$$\begin{aligned}
\varepsilon\frac{\partial u}{\partial \tau} &= \Delta_x u - k_1\varepsilon u, \quad x \in \Omega_1\backslash\Omega_\varepsilon \\
\partial_{nx} u &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon\partial_{nx} u &= \frac{u}{v+1}, \quad x \in \partial\Omega_{\varepsilon_1}
\end{aligned}$$

$$\begin{aligned}
\varepsilon\frac{\partial v}{\partial \tau} &= \Delta_x v - k_2 v, \quad x \in \Omega_1\backslash\Omega_\varepsilon \\
\partial_{nx} v &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon\partial_{nx} v &= u^2, \quad x \in \partial\Omega_{\varepsilon_2}.
\end{aligned}$$

We have already rescaled time with $\tau = \varepsilon t$.

As said in §2.1 when both α_u^2 and α_v^2 are $O(\varepsilon)$ then the outer and inner solutions all have $O(1)$ expansions. Since this is not the case here, the scaling of the inner and outer solutions depend on the particular choice of F and G. We expand the outer and inner solutions as in [61]

$$\begin{aligned}
u(x,\tau) &= \frac{1}{\sqrt{\varepsilon}}u_0(x,\tau) + \sqrt{\varepsilon}u_1(x,\tau) + \dots \\
u^{(i)}(y,\tau) &= \frac{1}{\sqrt{\varepsilon}}u_0^{(i)}(y,\tau) + \sqrt{\varepsilon}u_1^{(i)}(y,\tau) + \dots \\
v(x,\tau) &= v_0(x,\tau) + \varepsilon v_1(x,\tau) + \dots \\
v^{(i)}(y,\tau) &= \frac{1}{\varepsilon}v_0^{(i)}(y,\tau) + v_1^{(i)}(y,\tau) + \dots .
\end{aligned}$$

The analysis for u is the same as in §2.4 so that the ODE for $u_0(\tau)$ in (2.47) now becomes

$$\frac{\partial u_0}{\partial \tau} = -k_1 u_0 + 3\frac{u_0(\tau)}{v_0(x_1,\tau)}.$$

The major difference here is that v_0 is not only a function of τ but also of x. This is because the outer problems for v are

$$\Delta_x v_0 - k_2 v_0 = 0, \quad x \in \Omega_1\backslash\{x_2\}, \qquad \partial_{nx} v_0 = 0, \quad x \in \partial\Omega_1 \tag{2.52a}$$

$$\partial_\tau v_0 = \Delta_x v_1 - k_2 v_1, \quad x \in \Omega_1\backslash\{x_2\}, \qquad \partial_{nx} v_1 = 0, \quad x \in \partial\Omega_1. \tag{2.52b}$$

The inner problem for v_0 is

$$\Delta_y v_0^{(i)} = 0, \quad \rho > 1, \quad \partial_\rho v_0^{(i)} = -u_0^2, \quad \rho = 1.$$

From matching we determine that $v_0 \to 0$ as $\rho \to \infty$. From this as well as (2.4.1) we obtain

$$v_0^{(i)} = \frac{u_0^2}{\rho}.$$

Then we match and add in a delta function to (2.52a) to obtain,

$$\Delta_x v_0 - k_2 v_0 = -4\pi u_0^2 \delta(x - x_2), \quad x \in \Omega_1 \backslash \{x_2\}, \quad \partial_{nx} v_0 = 0, \quad x \in \partial\Omega_1. \tag{2.53}$$

The solution to (2.53) is

$$v_0(x, \tau) = 4\pi u_0^2 G_h(x; x_2; \sqrt{k_2}). \tag{2.54}$$

Here $G_h(x; x_0; \alpha)$ is the Modified Helmholtz Green's function which satisfies

$$\begin{aligned} &\Delta_x G_h - \alpha^2 G_h = -\delta(x - x_0), \quad x \in \Omega_1 \\ &\partial_{nx} G_h = 0, \quad x \in \partial\Omega_1 \,. \end{aligned}$$

The Green's function G_h can be written explicitly using standard methods (see [6]) and has the form

$$G_h(x; x_0; \alpha) = \frac{e^{-\alpha|x-x_0|}}{4\pi|x - x_0|} + G_R(x; x_0; \alpha),$$

where the regular part G_R is defined as

$$\begin{aligned} G_R(x; x_0; \alpha) &= \sum_{n=0}^{\infty} \frac{2n+1}{4\pi\sqrt{r r_0}} P_n(\cos\gamma) \frac{a_{1,n}}{a_{2,n}} I_{n+1/2}(\alpha r) I_{n+1/2}(\alpha r_0) \\ \frac{a_{1,n}}{a_{2,n}} &= \frac{(\alpha R) K_{n+3/2}(\alpha R) - n K_{n+1/2}(\alpha R)}{(\alpha R) I_{n+3/2}(\alpha R) + n I_{n+1/2}(\alpha R)} \,. \end{aligned} \tag{2.55}$$

Above, $I_\nu(x)$ and $K_\nu(x)$ are the modified Bessel functions of order ν of the first and second kind respectively. The Legendre polynomial is denoted P_n, and its argument is defined as $\cos\gamma = \cos\theta\cos\theta_0 + \sin\theta\sin\theta_0\cos(\phi - \phi_0)$. Finally, (r_0, θ_0, ϕ_0) denotes the spherical coordinates of x_0.

In summary we have

$$\begin{aligned} \partial_\tau u_0 &= -k_1 u_0 + 3\frac{u_0(\tau)}{v_0(x_1, \tau)} \\ v_0(x, \tau) &= 4\pi u_0^2 G_h(x; x_2; \sqrt{k_2}). \end{aligned} \tag{2.56}$$

This can be simplified to

$$\partial_\tau u_0 = -k_1 u_0 + \frac{3}{4\pi u_0 G_h(x_1; x_2; \sqrt{k_2})},$$

which has the solution

$$u_0 = \sqrt{Ce^{-2k_1\tau} + \frac{3}{4\pi k_1 G_h(x_1; x_2; \sqrt{k_2})}},$$

where, C is a constant determined by an initial condition.

Therefore spatial terms are seen explicitly in the ODE for $u_0(\tau)$ in (2.56). This is in contrast to the system in (2.47) where the spatial terms are not present.

In some signalling systems the concentration gradients do decay exponentially and therefore our mathematical model would need to use O(1) decay rates. As already mentioned, when the decay rates are O(1), the analysis is more difficult. This is because the scaling of the outer and inner solutions are dependent on the choice of F and G functions. For some cases it is not clear what the scaling should even be [61]. Also, the solutions to the Modified Helmholtz equation depend on the Modified Helmholtz Green's function. The regular part of this Green's function, defined in (2.55) is more difficult to work with compared to the regular part of the Neumann Green's function defined in (2.17).

2.5 Examples

In this section we look at several examples of simple signalling pathways to show how the results from previous sections can be used and to compare them to numerical computations. The analysis in previous sections was done for signalling pathways with only two signalling proteins. We look at some examples in this section where there are more than two proteins and in these cases we can easily extend the results. For example, for a pathway with three proteins, we may say we are using the results from (2.47) which only has two equations. What we mean is we are using the obvious extension of (2.47) to a larger system.

In most of the examples in this section we compare asymptotic solutions with numerical solutions. We solve the PDE BVPs numerically using the software package Comsol [1]. The corresponding Comsol code, which was is used for the example in

§2.5.3, can be found in the Appendix in §A.2. When solving a system of ODEs we use the standard ODE solver package in *Maple* [2].

Comsol is a FEM package for solving PDEs defined over complicated geometries in one, two, and three dimensions. We use the time dependent solvers in Comsol to solve (2.3). Comsol has adaptive time stepping algorithms and there are many different ones to choose from. We use the default choice which is a family of backward differentiation formulas (BDF). A BDF formula is a multistep formula based on numerical differentiation for solutions to ODEs. A BDF method of order n computes the solution using an nth-grade polynomial in terms of backward differences. We also can choose relative and absolute tolerances in Comsol to control the accuracy of the solutions. For solving the models without delay we used 1e-6 for both the absolute and relative tolerances.

One other thing to mention is that Comsol does not have adaptivity in the spatial dimension. This means that the mesh is constant over the entire computation of the solution. The global mesh properties such as the number of elements and their sizes is determined by the geometry. The mesh is not adaptive. Although a user can go in and tweak the mesh properties within Comsol, we use the default mesh properties for our simulations.

In this chapter we use the standard ODE solvers in Maple to solve the ODE systems which approximate the PDE solutions. In Chapter 3 we use Matlab instead of Maple. In Maple we use the default initial value problem (IVP) solver. The default method for IVPs in Maple is a Runge-Kutta Fehlberg method that produces a fifth order accurate solution. This fourth-fifth order Runge-Kutta method is commonly abbreviated as rk45 [15]. We use the default tolerances in Maple which are specific to the method used. The default values for rkf45 are an absolute tolerance of 1e-7 and a relative tolerance of 1e-6. The value for the initial time step is determined by the method and takes into account the local behavior of the ODE system.

We did not encounter any major difficulties doing the numerical computations here because the ODEs and PDEs are quite standard. This is contrasted with the numerical simulations in Chapter 3 where delay is introduced. In that chapter the inclusion of delay complicates the numerical solutions and some innovation is required. Therefore we go into a lot more detail when discussing the numerical simulations in

2.5.1 Bistability

Positive feedback within a signalling pathway can lead to bistability but it is not a prerequisite for it [80]. It has been speculated that bistability plays a role in cell responses such as differentiation. Bistability can form switch like responses and can be viewed as producing a type of biochemical memory.

Here we consider a simplified cascade model where the boundary conditions in (2.3) are chosen to be Hill functions as used in the ODE model in [83]. The model is simplified because it neglects the active and inactive forms of the signalling proteins. We use a cascade with two cycles for simplicity although similar results could be obtained with a cascade of arbitrary length. This system has forward activation as well as positive feedback. Here S represents the stimulus from a cell surface receptor. We consider it to be a constant in the first part of this example. The model from (2.3) becomes

$$\begin{aligned}
\tau_u \frac{\partial u}{\partial t} &= \Delta_x u - \alpha_1^2 \varepsilon u, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} u &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon \partial_{nx} u &= \frac{(S+\eta v)^\beta}{\alpha^\beta + (S+\eta v)^\beta}, \quad x \in \partial\Omega_{\varepsilon_1} \\
\tau_v \frac{\partial v}{\partial t} &= \Delta_x v - \alpha_2^2 \varepsilon v, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} v &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon \partial_{nx} v &= \frac{u^\beta}{\alpha^\beta + u^\beta}, \quad x \in \partial\Omega_{\varepsilon_2}.
\end{aligned} \tag{2.57}$$

We choose the Hill coefficient $\beta = 2$ and all other parameter values as $\alpha = \sqrt{12}, \eta = 10/4, R = 1, k_1 = 3, Du = 4, k_2 = 3, Dv = 4, x_1 = (1/2, 0, 0), x_2 = (-1/2, 0, 0), \varepsilon = 0.01$ and treat S as a bifurcation parameter. To solve for the steady state of (2.57) we need to use the results from §2.2. We will only use the leading order results here because that is all that is needed to observe the bistability. The equilibrium solutions are constant to leading order and are determined by the system in (2.12).

The system of equations in (2.12) becomes

$$\begin{aligned} u_0 &= \frac{4(S+\frac{5}{2}v_0)^2}{12+(S+\frac{5}{2}v_0)^2} \\ v_0 &= \frac{4u_0^2}{12+u_0^2}. \end{aligned} \tag{2.58}$$

For different values of S we can solve (2.58) numerically to produce the following bifurcation diagram for u_0. There is a similar bifurcation diagram for v_0.

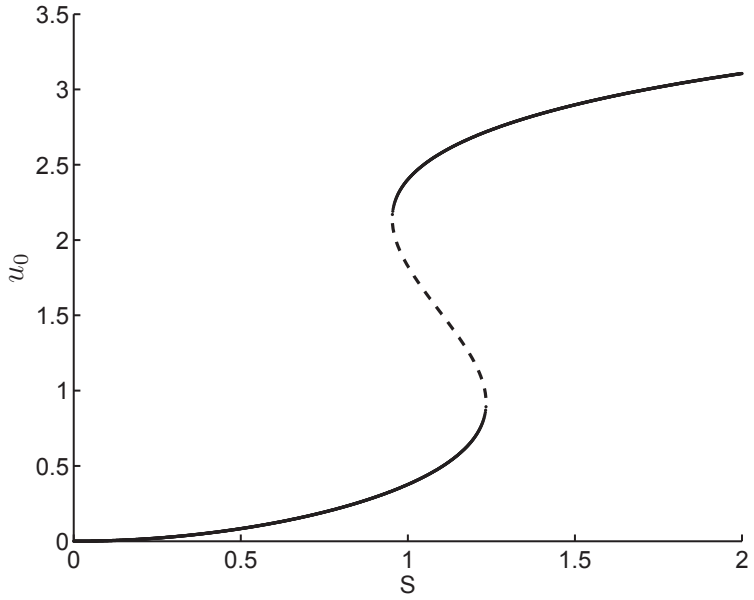


Figure 2.3: The steady state solution $u(x)$ is the constant u_0 to leading order. Here we plot u_0 as a function of S to show how bistability can arise for specific values of S. The dotted line is the unstable branch of equilibria and the solid line is the stable branch.

We now calculate the eigenvalues for the steady state solutions of (2.57) using the results from §2.3. We will do this for $S = 1.1$ because there is three steady state solutions in this case. We also compare the asymptotic eigenvalues with those obtained numerically in Comsol. The results are summarized in Table 2.1.

Table 2.1: Comparison of the numerical calculation of the first two principle eigenvalues ($\lambda_{i,ii}$) from Comsol with the asymptotic eigenvalues (including the correction term) from §2.3. The eigenvalue problem in (2.28) was solved numerically in Comsol to obtain the numerical approximations.

Leading Order Steady States and Corresponding Eigenvalues		
Steady State (u_0, v_0)	Numerical	Asymptotic
$(1.5127, 0.6406)$	$\lambda_{i,ii} = -0.06333, 0.00315$	$\lambda_{i,ii} = -0.06343, 0.00328$
$(0.4926, 0.0793)$	$\lambda_{i,ii} = -0.05168, -0.00840$	$\lambda_{i,ii} = -0.05170, -0.00836$
$(2.5781, 1.4258)$	$\lambda_{i,ii} = -0.05503, -0.00507$	$\lambda_{i,ii} = -0.05543, -0.00466$

Now we let the parameter S be a slowly varying function of time. In Figure 2.3, as S increases and crosses approximately $S = 0.95$ there is a change from the existence of a single stable equilibrium to the existence of two stable equilibrium and one unstable one. Then as S crosses approximately $S = 1.2$ the system has only one stable equilibrium again. We will choose S to be a piecewise function that increases slowly in time from zero to two, stays at two for a period of time, and then decreases slowly back to zero. We solve the BVP in (2.57) numerically (see Figure 2.4) with this choice of $S(t)$ to observe the switching mechanism caused by the bifurcations illustrated in Figure 2.3.

The solution for $u(x,t)$ is basically constant in space away from its point source x_1 (outer region). This was shown in §2.4. The approximation of u and v in their corresponding outer regions is governed by the ODEs in (2.47),

$$\begin{aligned} \tau_u \frac{\partial u_0}{\partial \tau} &= -\alpha_1^2 u_0 + 3\frac{(S+\eta v_0)^\beta}{\alpha^\beta + (S+\eta v_0)^\beta} \\ \tau_v \frac{\partial v_0}{\partial \tau} &= -\alpha_2^2 v_0 + 3\frac{u_0^\beta}{\alpha^\beta + u_0^\beta} \end{aligned} \tag{2.59}$$

Note that the ODE system reduction of (2.57) is just a rescaling of the equations used in [83]. In this case the inclusion of spatial effects does not effect the leading order dynamics. This is because $c_1 = F(v)$ and $c_2 = G(u)$ are both defined explicitly

in (2.47). This will always occur when the production functions contain no self regulation (i.e. $F_u = 0$ and $G_v = 0$).

We can compare the numerical computations of the PDE system in (2.57) with the ODE system in (2.47). The results are in Figure 2.4.

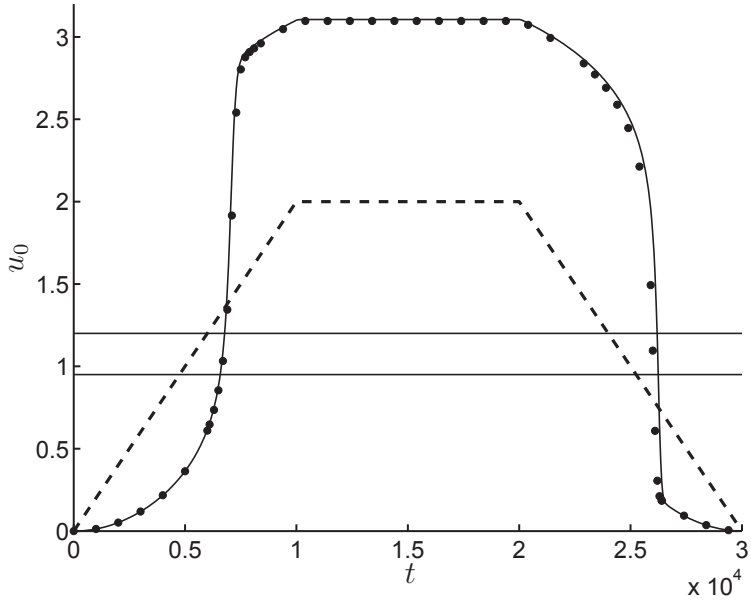


Figure 2.4: Here we plot the numerical solution of $u(x,t)$ (dotted curve) from (2.57) at the single point $(0.99, 0, 0)$. The solution at this point represents the value of u in the outer region away from the first compartment centred at $x_1 = (0.5, 0, 0)$. The solid curve is the asymptotic solution $u_0(\tau)$, which is an approximation to the solution $u(x,t)$ of (2.57) in the outer region. The leading order approximation, $u_0(\tau)$, was obtained by solving the ODE system in (2.47) numerically. There is good agreement between the simpler ODE model in (2.47) and the full 3D BVP simulations of (2.57). The dashed line is $S(t)$ and the two horizontal solid lines are the approximate bifurcation values $S = 0.95$, $S = 1.2$ from the bifurcation diagram in Figure 2.3.

As $S(t)$ crosses the bifurcation point $S = 1.2$, the solution jumps away from the now unstable equilibrium toward the stable one. This represents the signalling switch being turned on. Then as S decreases and crosses $S = .95$, the solution jumps back down to the only stable equilibrium and the switch is turned off.

2.5.2 Robust Switching Model with Delayed Bifurcation

Here we choose enzyme kinetic functions which may not be as biologically relevant as Hill or Michaelis Menten functions but they provide dynamics with a robust switch. We could use the more realistic functions in the same family as those considered in the previous section and get the same dynamic behavior. However if we adapted those production functions to a signalling cascade similar to the ones considered in this section (F and G both dependent on u and v) the analysis would not be straight forward and all unknowns would have to be calculated numerically. In particular, the leading order equilibria would be solutions to high degree polynomials, the eigenvalues would not be solved for explicitly, and it would be more difficult to obtain explicit expressions for the system of ODEs coming from (2.47). We have chosen these enzyme kinetic functions to resolve these issues and to show the construction of the system of ODEs governing the dynamics of the system. More importantly, with these simpler functions, we can demonstrate how multiple solutions for c_1 and c_2 in (2.47) are handled.

We first do the analysis for a pathway with two signalling proteins to show why there is a switch and then show numerical computations for a pathway with four proteins. The model from (2.3) becomes

$$\tau_u \frac{\partial u}{\partial t} = \Delta_x u - \alpha_1^2 \varepsilon u, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \tag{2.60a}$$

$$\partial_{nx} u = 0, \quad x \in \partial\Omega_1 \tag{2.60b}$$

$$\varepsilon \partial_{nx} u = k_1/3, \quad x \in \partial\Omega_{\varepsilon_1} \tag{2.60c}$$

$$\tau_v \frac{\partial v}{\partial t} = \Delta_x v - \alpha_2^2 \varepsilon v, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \tag{2.60d}$$

$$\partial_{nx} v = 0, \quad x \in \partial\Omega_1 \tag{2.60e}$$

$$\varepsilon \partial_{nx} v = -v(v - 11/6)(u - 1/2), \quad x \in \partial\Omega_{\varepsilon_2}, \tag{2.60f}$$

with parameter values $R = 1, D_u = 1, D_v = 1, k_1 = 1, k_2 = 1$. The reason for the particular choice of functions in (2.60c) and (2.60f) is so the equilibrium values work out nicely as well as to give the desired switch behavior. We will assume that $u(x, 0) = 0$ and $v(x, 0) = 0$ and we will determine the evolution of u and v to leading order.

We first use the result from (2.12) to find the leading order equilibrium solutions. In this example (2.12) becomes

$$u_0 = 1,$$
$$v_0 = -4v_0((4/3)v_0 - 11/6)(u_0 - 1/2).$$

The two leading order steady state solutions are the constant solutions $(u_0, v_0) = (1, 0)$ and $(u_0, v_0) = (1, 1)$. To determine the stability of the two equilibriums we must solve the eigenvalue problem in (2.35). For the equilibrium $(u_0, v_0) = (1, 0)$ the eigenvalues are $\lambda_{i,ii} = 32, -1$ so the equilibrium is unstable. For $(u_0, v_0) = (1, 1)$ they are $\lambda_{i,ii} = -1, -32/17$ so the equilibrium is stable. From this we know $u_0 \to 1$ and $v_0 \to 1$ as $t \to \infty$. We now comment on how the leading order solutions, $u_0(\tau)$ and $v_0(\tau)$, evolve before reaching equilibrium.

Since the boundary condition in (2.60c) does not depend on v we can analyze the BVP for u independently of v. Therefore we can treat u_0 as a bifurcation parameter. Although u_0 is a function of time we will first consider the static case where u_0 is a constant. If u_0 is constant then the leading order equilibrium solutions are $v_0 = 0$ and $v_0 = \frac{-7+11u_0}{8u_0-4}$, with corresponding eigenvalues $\lambda = \frac{-8(-7+11u_0)}{22u_0-23}$ and $\lambda = \frac{-8(-7+11u_0)}{22u_0-5}$ respectively. For $u_0 < 7/11$, $v_0 = 0$ is stable while the other equilibrium is unstable. This is reversed for $u_0 > 7/11$. Therefore, for values of $u_0 < 7/11$ the solution $v_0(\tau)$ is attracted to $v_0 = 0$ and for values of $u_0 > 7/11$ the solution $v_0(\tau)$ is attracted to the other equilibrium $v_0 = \frac{-7+11u_0}{8u_0-4}$. This same qualitative behavior is observed even when u_0 is a function of τ but with one main difference to be explained shortly.

From the ODE for $u_0(\tau)$ in (2.47) along with the zero initial condition already assumed, we have that $u_0(\tau) = 1 - e^{-\tau}$. We can treat $u_0(\tau)$ as a time dependent bifurcation parameter. The solution for $v_0(\tau)$ is initially attracted to $v_0 = 0$ as described above in the static case. Then we expect it to be repelled from $v_0 = 0$ as $u_0(\tau)$ crosses through the critical value $u_0 = 7/11$. What actually happens is that $v_0(\tau)$ remains attracted to $v_0 = 0$ for a period of time after $u_0(\tau)$ crosses through the critical point, before being repelled away from $v_0 = 0$ and attracted to $v_0 = 1$. This phenomenon where the change of stability expected at a bifurcation is delayed due to the time dependence of a bifurcation parameter, is known as a delayed bifurcation. It is usually the result of the bifurcation parameter slowly evolving in time [71].

The switching behavior in the solution for $v_0(\tau)$ from $v_0 = 0$ to $v_0 = 1$, along with the delayed bifurcation, can be seen in Figure 2.5. Here we solve the ODE system in (2.47) numerically applied to the current example.

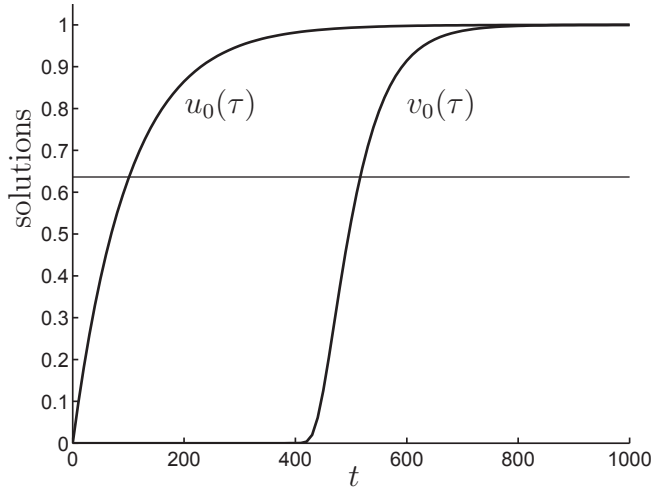


Figure 2.5: Here we plot the numerical solution to the ODE system in (2.47) for the particular choice of F and G from the BVP in (2.60). The solutions $u_0(\tau)$ and $v_0(\tau)$ represent how the solutions to (2.60) change in their corresponding outer regions. As u_0 crosses $u_0 = 7/11$ (shown as horizontal solid line) the equilibrium $v_0 = 0$ becomes unstable and $v_0(\tau)$ is attracted to the stable equilibrium $v(\tau) = 1$. The jump in the solution for v_0 from $v_0 = 0$ to $v_0 = 1$ does not occur immediately as u_0 crosses the bifurcation point. This is because there is a delayed bifurcation.

For this example, we will construct (2.47) explicitly. We only need to find the ODE for $v_0(\tau)$, since the ODE for $u_0(\tau)$ is already known. We first find the solution to $c_2(\tau) = G(u_0(\tau), v_0(\tau) + c_2(\tau))$ from (2.47). This results in a quadratic equation for $c_2(\tau)$ with roots

$$c_2^{\pm} = \frac{-23 - 24v_0(\tau)u_0(\tau) + 12v_0(\tau) + 22u_0(\tau)}{12(2u_0(\tau) - 1)} \pm \frac{\sqrt{529 + 576v_0(\tau)u_0(\tau) - 288v_0(\tau) - 1012u_0(\tau) + 484u_0(\tau)^2}}{12(2u_0(\tau) - 1)}.$$

We choose the c_2^+ branch in defining the ODE for $v_0(\tau)$ because it results in the same leading order equilibrium points, $(u_0, v_0) = (1, 0)$ and $(u_0, v_0) = (1, 1)$, found earlier in this section for the full PDE system. Another remark is that the two branches,

$c_2^{\pm}$ do not intersect in the region of interest $0 \leq u_0, v_0 \leq 1$. The ODE for $v_0(\tau)$ is

$$\frac{\partial v_0(\tau)}{\partial \tau} = -v_0(\tau) + 3c_2^{+}.$$

We can extend the model in (2.60) by adding in two more variables w and z. We use the same parameter values for the cell radius, decay rates, and diffusion coefficients as above. We use the same form of boundary condition in (2.60f) but for w it becomes $-w(w - 11/6)(v - 1/2)$ while for z it is $-z(z - 11/6)(w - 1/2)$. The compartment locations for u, v, w, and z are now $(0.75, 0, 0)$, $(0.25, 0, 0)$, $(-0.25, 0, 0)$, and $(-0.75, 0, 0)$ respectively. The initial conditions are $u(x, 0) = 0$, $v(x, 0) = 0.05$, $w(x, 0) = 0.05$, and $z(x, 0) = 0.05$. As in the system with two variables, we must choose the correct branches for each c_i. For example, associated with the ODE for $w_0(\tau)$ there will be a $c_3^{\pm}$ which will have the same expression as $c_2^{\pm}$ but with $v_0(\tau)$ replaced by $w_0(\tau)$ and $u_0(\tau)$ replaced by $v_0(\tau)$. We choose the c_2^{+} branch again for the same reason as before. The same scenario holds for $z_0(\tau)$ and $c_4^{\pm}$.

We solved the leading order ODE system for the two variable example in Figure 2.5. We now solve the full PDE model in (2.60) (as well as the ODE reduction (2.47)) but extended to a system of four variables. For each solution of the PDE system we plot it at a single point in its corresponding outer region as seen in Figure 2.6. We have chosen $\varepsilon = 0.05$ so the agreement between the full PDE and approximating asymptotic ODE model will not be as good when $\varepsilon = 0.01$. Also, our asymptotic solutions are valid when the solutions of the PDE model are not changing too quickly. In the current example the solutions are changing quickly from one equilibrium to another.

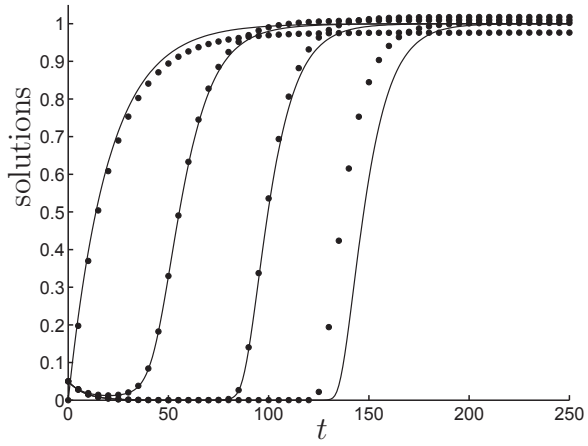


Figure 2.6: The four curves (dotted) from left to right are u , v, w , and z plotted in their respective outer regions. These solutions are found from solving the four variable extension of (2.60) numerically. The solutions for u and v were plotted at the point (-0.99,0,0) and w and z were plotted at (0.99,0,0). Each solution besides u is initially attracted toward zero but then jumps away to the other equilibrium because of a change in stability. The solid lines are the corresponding asymptotic solutions from solving the four variable extension of (2.47).

2.5.3 Hopf Bifurcation

In this section we look at a pathway with three signalling proteins each of which can be in an active and inactive form. We assume that the total concentrations are $u_T = v_T = w_T = 1$. We use Hill functions as in [83] but we also add in the deactivated form of the signalling proteins as a linear multiplier (see equations (2.61c), (2.61f),

(2.61i)). The model in (2.3) for three variables then becomes

$$\tau_u \frac{\partial u}{\partial t} = \Delta_x u - \alpha_1^2 \varepsilon u, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \tag{2.61a}$$

$$\partial_{nx} u = 0, \quad x \in \partial\Omega_1 \tag{2.61b}$$

$$\varepsilon \partial_{nx} u = \frac{S^3}{S^3 + (\alpha + \eta w)^3}(1 - u), \quad x \in \partial\Omega_{\varepsilon_1} \tag{2.61c}$$

$$\tau_v \frac{\partial v}{\partial t} = \Delta_x v - \alpha_2^2 \varepsilon v, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \tag{2.61d}$$

$$\partial_{nx} v = 0, \quad x \in \partial\Omega_1 \tag{2.61e}$$

$$\varepsilon \partial_{nx} v = \frac{u^3}{\alpha^3 + u^3}(1 - v), \quad x \in \partial\Omega_{\varepsilon_2} \tag{2.61f}$$

$$\tau_w \frac{\partial w}{\partial t} = \Delta_x w - \alpha_3^2 \varepsilon w, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \tag{2.61g}$$

$$\partial_{nx} w = 0, \quad x \in \partial\Omega_1 \tag{2.61h}$$

$$\varepsilon \partial_{nx} w = \frac{v^3}{\alpha^3 + v^3}(1 - w), \quad x \in \partial\Omega_{\varepsilon_3}, \tag{2.61i}$$

where we choose $D_u = 1/3, D_v = 1/3, D_w = 1/3, R = 1, k_1 = 1/2, k_2 = 1, k_3 = 1/4, \alpha = 0.2, \eta = 50, x_1 = (1/2, 0, 0), x_2 = (0, 0, 0), x_3 = (-1/2, 0, 0), \varepsilon = 0.01$, and treat the stimulus S as a bifurcation parameter.

We use the extended version of the system of differential equations derived in (2.47) to get an approximation for how the solutions away from the compartments evolve in time. Recall that the solutions away from the compartments are basically constant with respect to space. The system of ODEs from (2.47) applied to three variables becomes

$$\begin{aligned}
\frac{\partial u_0}{\partial \tau} &= -\frac{1}{2} u_0 + \frac{S^3}{2S^3 + (\alpha + \eta w_0)^3}(1 - u_0) \\
\frac{\partial v_0}{\partial \tau} &= -v_0 + \frac{u_0^3}{\alpha^3 + 2u_0^3}(1 - v_0) \\
\frac{\partial w_0}{\partial \tau} &= -\frac{1}{4} w_0 + \frac{v_0^3}{\alpha^3 + v_0^3}(1 - w_0).
\end{aligned} \tag{2.62}$$

The three equations which determined c_1, c_2, and c_3 from the three variable extension of (2.47) were linear in c_1, c_2, and c_3 respectively. Therefore we did not have to consider multiple solution branches as in §2.5.2.

From solving (2.62) numerically we observe that there is a Hopf bifurcation and as a result there are values of S which lead to sustained oscillations in the concentration of the signalling proteins. For different values of S we can make the following Hopf bifurcation diagram for u_0. There are similar diagrams for v_0 and w_0.

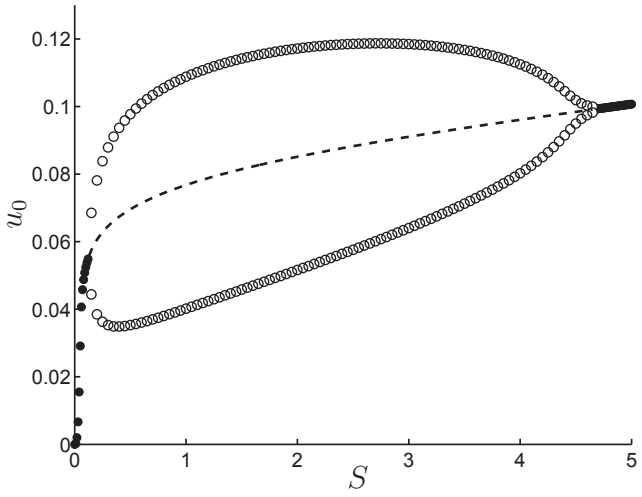


Figure 2.7: The steady state solution $u(x)$ is the constant u_0 to leading order. Here we plot u_0 as a function of S as well as the max and min (open circles) of sustained oscillations near the Hopf Bifurcation points. The solid circles represent the stable equilibrium points which then become unstable (dotted line) as S crosses the first bifurcation point. As a result there are sustained oscillations about the unstable equilibrium. There is then another Hopf bifurcation where the unstable equilibrium become stable again.

The above diagram was made by solving the simpler ODE system in (2.62) numerically. This system has come out of the analysis from §2.4 and approximates the dynamics of the leading order solutions from the more complicated BVP model in (2.61). To see how well the ODE model approximates the PDE model we can solve both numerically and compare them. We will plot the solution from (2.61) at one point in space in the outer region for u_0. The result is in Figure 2.8 and there is good agreement between the two models. It is important to point out that the Hopf bifurcation was found first in the ODE model (2.62) which can be solved numerous times very quickly compared to the PDE model. Then the full PDE model is solved numerically and verifies that the Hopf bifurcation occurs in that system as well.

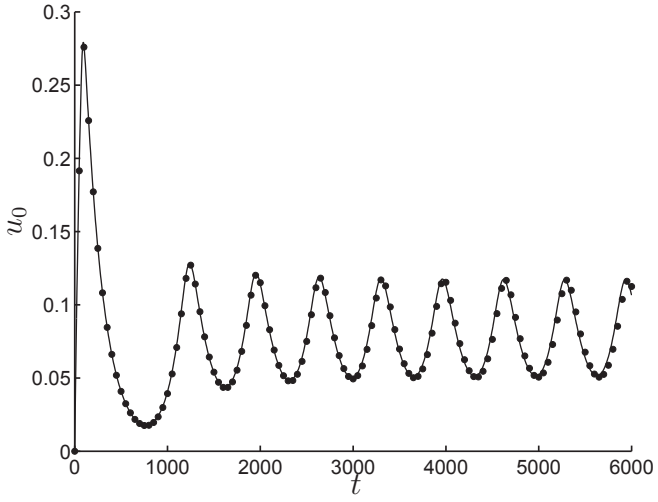
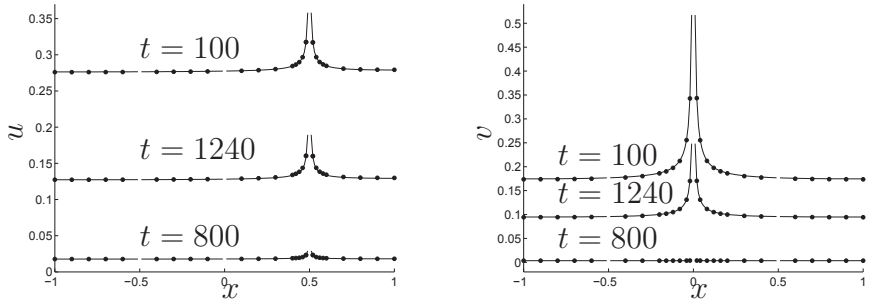


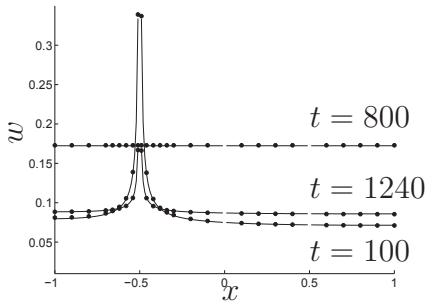
Figure 2.8: The dotted line is obtained from plotting the solution $u(x,t)$ from the full BVP model in (2.61) at the single point $(-0.99, 0, 0)$. It represents how $u(x,t)$ evolves in the outer region away from the compartment Ω_{ε_1} which is centred at $x_1 = (1/2, 0, 0)$. The solid line represents the solution $u_0(\tau)$, where $\tau = \varepsilon t$, which is found from solving the ODE model in (2.62). Here $S = 2$.

Thus far when we have used the results from §2.4 we have only used the leading order ODE system in (2.47). We now take into account the second order calculations which derived the system of ODEs in (2.50). Solving the system in both (2.47) and (2.50) numerically allows us to find the time and space dependent solutions in (2.51). We compare the solutions from (2.51) with the full BVP model (2.61) in Figure 2.9.

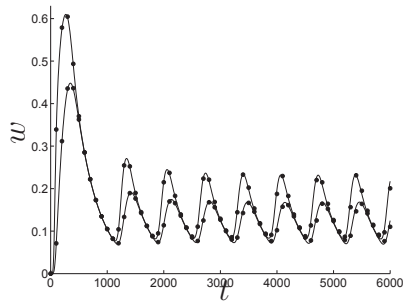


(a) $u(x,t)$

(b) $v(x,t)$



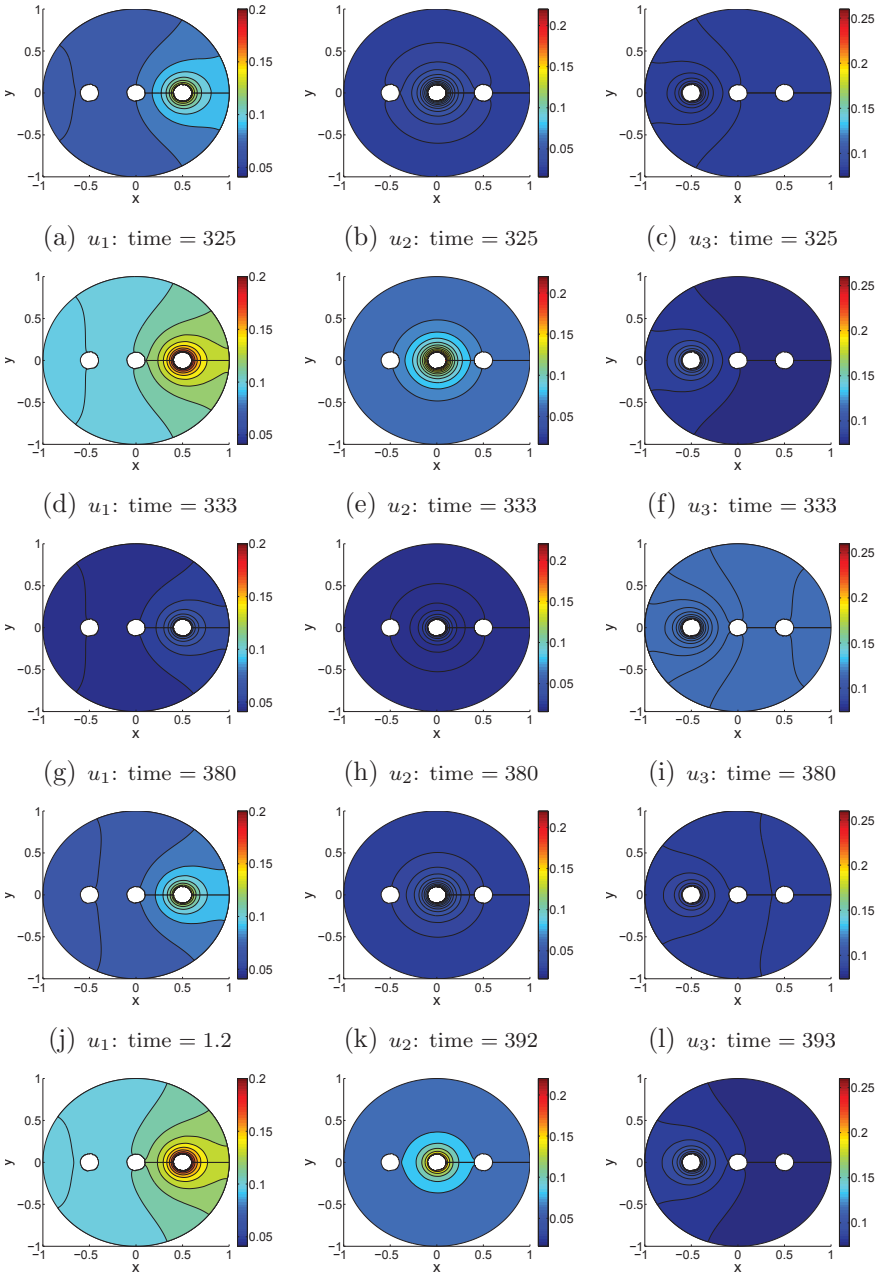
(c) $w(x,t)$



(d) $w(t)$ plotted as function of t at the point (0.99,0,0) and then at (-0.51,0,0).

Figure 2.9: In 2.9(a), 2.9(b), and 2.9(c) we plot the numerical solutions u, v, and w (dots) from BVP (2.61) along with their asymptotic solutions (solid line) from (2.51) for different time values. Each asymptotic solution from (2.51) depends on two functions which are each determined by different systems of ODEs. The functions $u_0(\tau)$, $v_0(\tau)$, and $w_0(\tau)$ were found from solving (2.47) numerically and the functions $\chi_1(\tau)$, $\chi_2(\tau)$, and $\chi_3(\tau)$ were found from solving (2.50) numerically. In 2.9(d) we plot the numerical solution for $w(x,t)$ from (2.61) at a point in its outer region, (.99,0,0), as well as a point in its inner region, (-0.51,0,0). The one with the higher amplitude is the one in the inner region. We then plot the asymptotic solutions from (2.51) at these same points. Note that for certain time values the height of the outer and inner region solutions are the same. This is why the solution for w in 2.9(c) at $t = 800$ appears flat across the domain. We are using $S = 2$ so there is sustained oscillations occurring in all these figures.

We end this Hopf bifurcation example with a time series evolution of the three variables. We use cross section plots here which are different from all the plots seen thus far. The sustained oscillations can be observed but more importantly we include these plots because they show more of the 3D structure of the solutions. The solution values are the numerical solutions from the PDE system (2.61).



(a) u_1: time = 325 (b) u_2: time = 325 (c) u_3: time = 325

(d) u_1: time = 333 (e) u_2: time = 333 (f) u_3: time = 333

(g) u_1: time = 380 (h) u_2: time = 380 (i) u_3: time = 380

(j) u_1: time = 1.2 (k) u_2: time = 392 (l) u_3: time = 393

(m) u_1: time = 402 (n) u_2: time = 402 (o) u_3: time = 402

Figure 2.10: Time series evolution of (2.61) where the solutions are plotted as cross sections through the x-y plane. Here we use $\varepsilon = 0.1$ and $S = 2$ and all other parameters values as in this section.

2.5.4 Comparison of (2.47) with other ODE Models

In this example we highlight the differences between the approximating ODE model in (2.47) with other ODE systems that are well mixed and which neglect spatial

effects. Throughout this chapter we have assumed linear deactivation everywhere in the cell and localized activation (modelled by the functions F and G) at subcellular compartments for the enzyme kinetics. A typical ODE model with the same enzyme kinetics but where activation occurs everywhere in the cell can be expressed as

$$\begin{aligned} \frac{\partial u}{\partial \tau} &= -u + F(u,v) \\ \frac{\partial v}{\partial \tau} &= -v + G(u,v). \end{aligned} \tag{2.63}$$

For simplicity, throughout this example we are assuming $\tau_u = 3$, $\tau_v = 3$, $k_1 = k_2 = 1$,and $u_T = v_T = 1$.

The signalling model considered in this chapter takes into account spatial effects, diffusion, and enzyme localization to subcellular sites. The system was modelled by the PDEs in (2.3). The leading order dynamics of (2.3) are governed by the DAEs in (2.47) which becomes

$$\begin{aligned} \frac{\partial u_0}{\partial \tau} &= -u_0 + c_1(u_0, v_0), \quad c_1 = F(u_0 + c_1, v_0) \\ \frac{\partial v_0}{\partial \tau} &= -v_0 + c_2(u_0, v_0), \quad c_2 = G(u_0, v_0 + c_2). \end{aligned} \tag{2.64}$$

This system of DAEs can be converted to a system of two ODEs as long as the nonlinear equations can be solved for c_1 and c_2 in terms of u_0 and v_0.

The difference between (2.63) and (2.64) is how the activation functions F and G appear on the right hand side of the ODEs. When activation occurs everywhere, F and G appear in the ODEs explicitly as in (2.63). When spatial effects are considered and activation is localized, the activation functions F and G appear in the implicitly defined relationships $c_1 = F(u_0 + c_1, v_0)$ and $c_2 = G(u_0, v_0 + c_2)$. This results in a different form of nonlinearity.

For example, consider the following choice of F and G which have the same form as those used in [47],

$$\begin{aligned} F(u,v) &= \frac{1-u}{(1-u+\alpha)(1+\eta v)} \\ G(u,v) &= \frac{(1-v)u}{1-v+\alpha}. \end{aligned}$$

For this choice of F and G, c_1 and c_2 are the solutions to the following quadratics

$$\begin{aligned} (1+\eta v_0)c_1^2 - ((1+\eta v_0)(\alpha + 1 - u_0) + 1)c_1 + 1 - u_0 = 0 \\ c_2^2 - (\alpha + 1 - v_0 + u_0)c_2 + u_0(1 - v_0) = 0. \end{aligned} \tag{2.65}$$

Since the the equations in (2.65) are quadratic, c_1 and c_2 will both have radicals in their expressions. This is a different type of nonlinearity than that of F and G in (2.63). This is one way in which the ODEs in (2.64) differ from those in (2.63). The systems will have the same number of equilibrium points, but the dynamical behavior could be different. Secondly, since there are two possible solutions for c_1 and two possible solutions for c_2, which we will denote as $c_1^{\pm}$ and $c_2^{\pm}$ respectively, me must go through all possible choices of solution branches. As already mentioned in previous sections, we must choose the correct two solution branches (one for c_1 and one for c_2) which are defined in the region of interest and give the correct dynamical behavior. For this example, if we choose c_1^- and c_2^- then the solutions of (2.64) have the same equilibrium points as the leading order equilibrium solutions of the full PDE system in (2.3). Therefore these are the correct solution branches to choose. No other pair of solution branches will give this same result.

Another difference between (2.64) and (2.63) is that the right hand side of (2.64) may become complex. As a very simple example, consider a system with one signalling protein and $F(u) = \frac{1}{4}u^2$ where we ignore the active and inactive forms as in [83]. The steady state solutions to leading order of the PDE system (2.3) are the constants $u_0 = 1$ (unstable) and $u_0 = 0$ (stable) which are found from solving (2.12). For the asymptotic ODE reduction, $c_1(u_0)$ satisfies $4c_1 = (c_1+u_0)^2$ and (2.64) becomes

$$\frac{\partial u_0}{\partial \tau} = 2 - 2u_0 - 2\sqrt{1-u_0}. \tag{2.66}$$

Note that we had two solution branches to choose from for c_1 and we chose the branch which results in (2.66) having the equilibrium solutions $u_0 = 0$ and $u_0 = 1$. Since we are ignoring the active and inactive forms we do not have any relationship which gives a restriction on the size of u_0. So for any initial condition $u_0(0) > 1$, the right hand side of the ODE in (2.66) is complex. The right hand side of the ODE in (2.63) is defined for any initial condition in this example as it becomes

$$\frac{\partial u}{\partial \tau} = -u + \frac{1}{4}u^2. \tag{2.67}$$

The equilibrium solutions of (2.67) are $u = 0$ (stable) and $u = 4$ (unstable). If an initial condition $u(0) > 4$ is chosen for (2.67) then the solutions blow up in a finite time.

An initial condition $u_0(0) > 1$ leads to (2.66) being complex. Thus the asymptotic ODE reduction is not defined for some choices of initial conditions. A reasonable question is to ask what this corresponds to in the full PDE system in (2.3). For this specific case of one variable u and $F = u^2/4$ we attempted running numerical simulations of (2.3) in Comsol for the initial condition $u(x, 0) = 1.1$. Since the asymptotic model is not defined for $u_0 > 1$ we wanted to see what behavior was evident in the full PDE system. We found that Comsol failed to converge and after reducing to a time step of $O(10^{-18})$ stopped the calculation due to a singularity. We believe the reason for this is that the solution is approaching infinity in a finite time (i.e. finite time blow up just as in (2.67)). The simulations for $0 < u(x, 0) < 1$ behaved as expected.

We have already mentioned briefly in §2.5.1, that the two systems (2.63) and (2.64) can have very similar right hand sides when $\frac{\partial}{\partial u}F(u, v) = 0$ and $\frac{\partial}{\partial v}G(u, v) = 0$. In this case the implicit relationships defined in (2.64) which determine c_1 and c_2, reduce to $c_1 = F(v_0)$ and $c_2 = G(u_0)$ respectively. With the choice of parameters we have chosen for this section, (2.63) and (2.64) are identical. In general, for arbitrary parameters, when F is independent of u and G is independent of v then the systems (2.47) and the ODE analog (without space) will agree up to scaling.

One final remark we would like to make is on the generality of (2.3). Since the enzyme kinetic functions F and G were chosen in such an arbitrary manner it appears we could construct any type of dynamical behavior (or any right hand side of (2.64)) that we want. Suppose we wanted the ODE reduction equations in (2.47), which govern the leading order dynamics of (2.3), to be

$$\begin{aligned} \frac{\partial u_0}{\partial \tau} &= -u_0 + \bar{f}(u_0, v_0) \\ \frac{\partial v_0}{\partial \tau} &= -v_0 + \bar{g}(u_0, v_0), \end{aligned} \tag{2.68}$$

for a specific choice of $\bar{f}$ and $\bar{g}$. This is easy in the well mixed systems (2.63) because we just choose $F = \bar{f}$ and $G = \bar{g}$. However, in order to turn (2.64)) into (2.68) we must find an F and G such that

$$\begin{aligned} \bar{f}(u_0, v_0) &= F(u_0 + \bar{f}(u_0, v_0), v_0) \\ \bar{g}(u_0, v_0) &= G(u_0, v_0 + \bar{g}(u_0, v_0)). \end{aligned}$$

In general it may not be possible to solve this functional relationship. Therefore, because of the implicit relationships defined in (2.64), we can not construct (at least easily) any type of dynamical behavior we want.

We leave the discussion of the results in this chapter to the last chapter in this thesis, Chapter 5.

Chapter 3

A Model with Delayed Enzyme Activation

In this chapter we analyze a 3D model which incorporates time delay explicitly. We use the Comsol and Matlab scripting languages along with the method of steps [8] to find numerical solutions of the full 3D PDE model with delay. We also use asymptotic analysis to find the stability of the equilibrium solutions. Asymptotic expressions for the solutions of the full model in time and space are obtained as well. Through this analysis the PDE model is approximated by a simpler system of delayed differential algebraic equations (DDAEs). Finally, because of the delay, we find that the systems under consideration may undergo Hopf bifurcations. In these cases sustained oscillations are observed and we use the Poincaré-Lindstedt method to improve upon the asymptotic approximations. This is an interesting application of a standard asymptotic method to a non standard problem.

The main sections of this chapter are as follows. In §3.1 we rewrite the main model from Chapter 2 and then add a constant time delay to get the model which will be analyzed in this chapter. In §3.2 we do a stability analysis to derive a nonlinear eigenvalue problem which determines the stability of the equilibrium solutions. In §3.3 we find approximate time dependent solutions of the model. Also in §3.3, the system of PDEs is approximated by a system of DDAEs. In §3.4 we discuss the numerical methods we have implemented, as well as associated methods, to solve the PDE model with delay as well as the DDAEs. In §3.6 we use the Poincaré-Lindstedt method to improve the analysis in the case of a Hopf bifurcation. We also compare numerical results with asymptotic results in §3.5 and §3.6.

3.1 Model

The following model was derived in Chapter 2 and can be found in equation (2.3),

$$
\begin{aligned}
\tau_u \frac{\partial u}{\partial t} &= \Delta_x u - \alpha_1^2 \varepsilon u, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} u &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon \partial_{nx} u &= F(u, v), \quad x \in \partial\Omega_{\varepsilon_1} \\
\\
\tau_v \frac{\partial v}{\partial t} &= \Delta_x v - \alpha_2^2 \varepsilon v, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} v &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon \partial_{nx} v &= G(u, v), \quad x \in \partial\Omega_{\varepsilon_2}.
\end{aligned}
\tag{3.1}
$$

We will now add a constant time delay to the model in (3.1). This is because we want the delay to affect the enzyme kinetic reactions that are responsible for activating the signalling molecules. These reactions take place on the boundaries of the internal compartments within the cell. Therefore we add the delay to the corresponding boundary conditions in (3.1). This means the enzyme kinetic functions F and G will take the form $F(u(x, t-s), v(x, t-s))$ and $G(u(x, t-s), v(x, t-s))$ for some constant delay s.

With the delay term only present in the boundary conditions, it allows for a simpler implementation for solving the PDE system with delay numerically (see §3.4). There are other options for where to add the delay in (3.1). We could add the delay to any of the terms in the PDE itself but this would lead to further numerical difficulties (which we may consider in future work). Moving the delay to a different term in (3.1) does not make the asymptotic analysis more complicated. However, it greatly complicates the methods we could use to compute approximate numerical solutions.

The model in (3.1) with a constant delay s now becomes,

$$\begin{aligned}
\tau_u \frac{\partial u}{\partial t} &= \Delta_x u - \alpha_1^2 \varepsilon u, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} u &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon \partial_{nx} u &= F(u(x, t-s), v(x, t-s)), \quad x \in \partial\Omega_{\varepsilon_1} \\
u(x,t) &= u_h(x,t), \quad t \leq 0 \\
\\
\tau_v \frac{\partial v}{\partial t} &= \Delta_x v - \alpha_2^2 \varepsilon v, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} v &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon \partial_{nx} v &= G(u(x, t-s), v(x, t-s)), \quad x \in \partial\Omega_{\varepsilon_2} \\
v(x,t) &= v_h(x,t), \quad t \leq 0.
\end{aligned} \tag{3.2}$$

Notice that we have added initial history data. Initial history is needed over one delay interval when one considers delay equations. This is contrasted with ODEs which only require initial conditions at a single point in time. Although the history functions u_h and v_h could be time and space dependent, for the examples later in this chapter, we will assume the initial history is constant in time and space.

Asymptotic approximations for the steady state solutions of (3.1) were found in Chapter 2 using the method of matched asymptotic expansions which will be used here as well. The addition of delay does not affect the steady state solutions. We rewrite them here for easy reference (see equations (2.27)),

$$\begin{aligned}
u_E(x) &= u_0 + \varepsilon \left(\frac{c_1}{|x - x_1|} + 4\pi c_1 R_n(x; x_1) + \chi_1 \right) + O(\varepsilon^2) \\
v_E(x) &= v_0 + \varepsilon \left(\frac{c_2}{|x - x_2|} + 4\pi c_2 R_n(x; x_2) + \chi_2 \right) + O(\varepsilon^2).
\end{aligned}$$

The constants u_0, v_0, c_1, and c_2 are found by solving the system of equations

$$\begin{aligned}
c_1 &= F(u_0 + c_1, v_0) \\
c_2 &= G(u_0, v_0 + c_2) \\
c_1 &= \frac{\alpha_1^2 u_0}{3} \\
c_2 &= \frac{\alpha_2^2 v_0}{3}.
\end{aligned} \tag{3.3}$$

This system is likely nonlinear (due to the nonlinearities in F and G) and the number of distinct solutions determines the number of equilibrium solutions of (3.2). There

is also a linear system which determines the constants χ_1 and χ_2 which can be found in (2.25) but we do not need to reference it in this chapter. The function $R_n(x, x_j)$ is the regular part of the Neumann Green's function defined in (3.22).

3.2 Stability Analysis

In Chapter 2 the stability of the equilibrium solutions of (3.1) were determined by a linear eigenvalue problem which can be found in (2.35). Because of the delay in (3.2), the stability of the equilibrium solutions will now be determined by a nonlinear eigenvalue problem. We will do a stability analysis and use matched asymptotics to analyze the resulting nonlinear eigenvalue problem. The method of matched asymptotic expansions will end up turning a complicated PDE eigenvalue problem, defined over a perturbed domain, into a simpler matrix eigenvalue problem.

We begin by introducing small perturbations off of the steady states solutions,

$$\begin{aligned} u &= u_E + \phi(x)e^{\lambda t}, \quad \phi \ll u_E \\ v &= v_E + \psi(x)e^{\lambda t}, \quad \psi \ll v_E, \end{aligned}$$

and substitute these into (3.2). After linearizing we get

$$\begin{aligned} \tau_u \lambda \phi &= \Delta_x \phi - \alpha_1^2 \varepsilon \phi, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\ \partial_{nx}\phi &= 0, \quad x \in \partial\Omega_1 \\ \varepsilon \partial_{nx}\phi &= e^{-\lambda s}\left(F_u(u_E, v_E)\phi + F_v(u_E, v_E)\psi\right), \quad x \in \partial\Omega_{\varepsilon_1} \\ \\ \tau_v \lambda \psi &= \Delta_x \psi - \alpha_2^2 \varepsilon \psi, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\ \partial_{nx}\psi &= 0, \quad x \in \partial\Omega_1 \\ \varepsilon \partial_{nx}\psi &= e^{-\lambda s}\left(G_u(u_E, v_E)\phi + G_v(u_E, v_E)\psi\right), \quad x \in \partial\Omega_{\varepsilon_2}. \end{aligned} \tag{3.4}$$

The eigenvalue problem is nonlinear because of the $e^{-\lambda s}$ terms in the boundary conditions. We will use the method of matched asymptotic expansions to analyze (3.4) and solve for λ.

It turns out that the delay has to be relatively large, $s = O(\frac{1}{\varepsilon})$, for it to have a significant affect on the dynamics of (3.2). We show this in §3.2. If the delay is smaller, $s = O(1)$ for example, then the qualitative behavior of (3.2) is similar to the

model without delay in (3.1). A delay of magnitude $s = O(1/\varepsilon)$ is required because of the $O(\varepsilon)$ scaling chosen for the decay parameters in (3.2).

Stability Analysis for Large Delay

In this section we find asymptotic approximations of (3.4) when $s = O(1/\varepsilon)$. The main result of this stability analysis is a nonlinear matrix eigenvalue problem which determines the stability of the steady state solutions of (3.2) to leading order.

We begin by expanding the eigenfunctions and λ in terms of ε. For the delay we let $s = T/\varepsilon$ where $T = O(1)$. To determine the scaling of λ in terms of ε we integrate the PDEs in (3.4) over the entire domain and apply the Divergence theorem,

$$\begin{aligned}\int_{\Omega_1\backslash\Omega_\varepsilon}(\tau_u\lambda\phi + \alpha_1^2\varepsilon\phi)\,dV &= 4\pi\varepsilon e^{-\lambda s}(F_u(u_E,v_E)\phi + F_v(u_E,v_E)\psi)\Big|_{|x-x_1|=\varepsilon}\\ \int_{\Omega_1\backslash\Omega_\varepsilon}(\tau_v\lambda\psi + \alpha_2^2\varepsilon\psi)\,dV &= 4\pi\varepsilon e^{-\lambda s}(G_u(u_E,v_E)\phi + G_v(u_E,v_E)\psi)\Big|_{|x-x_2|=\varepsilon}.\end{aligned} \tag{3.5}$$

These equations are useful in determining the correct order of the asymptotic expansions. For the method of matched asymptotics, both sides of (3.5) must be the same order of magnitude. This will work if λ is chosen to be $O(\varepsilon)$ since both sides of (3.5) would then be $O(\varepsilon)$.

We expand the outer eigenfunctions and λ in terms of ε as follows,

$$\begin{aligned}\phi(x) &= \phi_0(x) + \varepsilon\phi_1(x) + \varepsilon^2\phi_2(x) + \ldots\\ \psi(x) &= \psi_0(x) + \varepsilon\psi_1(x) + \varepsilon^2\psi_2(x) + \ldots,\end{aligned} \tag{3.6}$$

$$\lambda = \lambda_1\varepsilon + \lambda_2\varepsilon^2 + \ldots. \tag{3.7}$$

Next, we substitute the expansions from (3.6) and (3.7) into (3.4). After collecting like powers of ε we obtain the following outer problems,

$$\begin{aligned}
&0 = \Delta_x\phi_0, \quad x \in \Omega_1\backslash\{x_1\}, && \partial_{nx}\phi_0 = 0, x \in \partial\Omega_1\\
&0 = \Delta_x\phi_1 - (\alpha_1^2 + \tau_u\lambda_1)\phi_0, \quad x \in \Omega_1\backslash\{x_1\}, && \partial_{nx}\phi_1 = 0, x \in \partial\Omega_1\\
&0 = \Delta_x\phi_2 - (\alpha_1^2 + \tau_u\lambda_1)\phi_1 - \tau_u\lambda_2\phi_0, \quad x \in \Omega_1\backslash\{x_1\}, && \partial_{nx}\phi_2 = 0, x \in \partial\Omega_1\\
&0 = \Delta_x\psi_0, \quad x \in \Omega_1\backslash\{x_2\}, && \partial_{nx}\psi_0 = 0, x \in \partial\Omega_1\\
&0 = \Delta_x\psi_1 - (\alpha_2^2 + \tau_v\lambda_1)\psi_0, \quad x \in \Omega_1\backslash\{x_2\}, && \partial_{nx}\psi_1 = 0, x \in \partial\Omega_1\\
&0 = \Delta_x\psi_2 - (\alpha_2^2 + \tau_v\lambda_1)\psi_1 - \tau_v\lambda_2\psi_0, \quad x \in \Omega_1\backslash\{x_2\}, && \partial_{nx}\psi_2 = 0, x \in \partial\Omega_1.
\end{aligned} \tag{3.8}$$

To obtain the inner problems we introduce the rescaled inner variable $y = (x - x_j)/\varepsilon$ into (3.4). With the PDE and boundary conditions for ϕ specified on the compartment Ω_{ε_1}, we use $j = 1$. Similarly, with the PDE and boundary conditions for ψ we use $j = 2$. After changing variables in (3.4) from x to y we obtain,

$$\begin{aligned}
&\tau_u \lambda \phi^{(i)} \varepsilon^2 = \Delta_y \phi^{(i)} - \alpha_1^2 \varepsilon^3 \phi^{(i)}, \quad \rho > 1 \\
&\partial_{ny} \phi^{(i)} = e^{-\lambda s} \left(F_u(u_E, v_E)\phi^{(i)} + F_v(u_E, v_E)\psi \right), \quad \rho = 1 \\
&\\
&\tau_v \lambda \psi^{(i)} \varepsilon^2 = \Delta_y \psi^{(i)} - \alpha_2^2 \varepsilon^3 \psi^{(i)}, \quad \rho > 1 \\
&\partial_{ny} \psi^{(i)} = e^{-\lambda s} \left(G_u(u_E, v_E)\phi + G_v(u_E, v_E)\psi^{(i)} \right), \quad \rho = 1.
\end{aligned} \tag{3.9}$$

We expand the inner functions as

$$\begin{aligned}
\phi^{(i)}(y) &= \phi_0^{(i)}(y) + \varepsilon \phi_1^{(i)}(y) + \varepsilon^2 \phi_2^{(i)}(y) + \dots \\
\psi^{(i)}(y) &= \psi_0^{(i)}(y) + \varepsilon \psi_1^{(i)}(y) + \varepsilon^2 \psi_2^{(i)}(y) + \dots,
\end{aligned} \tag{3.10}$$

and substitute these into (3.9). The exponential term in the inner boundary conditions from (3.9) becomes

$$e^{-\lambda s} = e^{-(\varepsilon\lambda_1 + \varepsilon^2\lambda_2 + \dots)(T/\varepsilon)} = e^{-\lambda_1 T}(1 - \lambda_2 T\varepsilon + O(\varepsilon^2)).$$

After collecting powers of ε we get the following inner problems for $\phi^{(i)}$ and $\psi^{(i)}$

$$\begin{aligned}
0 &= \Delta_y \phi_0^{(i)}, \quad \rho > 1 \\
-\partial_\rho \phi_0^{(i)} &= e^{-\lambda_1 T} \left(F_u \phi_0^{(i)} + F_v \psi_0 \right), \quad \rho = 1 \\
0 &= \Delta_y \phi_1^{(i)}, \quad \rho > 1 \\
-\partial_\rho \phi_1^{(i)} &= e^{-\lambda_1 T} \left(F_u \phi_1^{(i)} + F_v \psi_1 + \phi_0^{(i)}(F_{uu} u_1^{(i)} + F_{uv} v_1) + \psi_0 (F_{uv} u_1^{(i)} + F_{vv} v_1) \right) \\
&\quad - \lambda_2 T e^{-\lambda_1 T} \left(F_u \phi_0^{(i)} + F_v \psi_0 \right), \quad \rho = 1 \\
0 &= \Delta_y \psi_0^{(i)}, \quad \rho > 1 \\
-\partial_\rho \psi_0^{(i)} &= e^{-\lambda_1 T} \left(G_u \phi_0 + G_v \psi_0^{(i)} \right), \quad \rho = 1 \\
0 &= \Delta_y \psi_1^{(i)}, \quad \rho > 1 \\
-\partial_\rho \psi_1^{(i)} &= e^{-\lambda_1 T} \left(G_u \phi_1 + G_v \psi_1^{(i)} + \phi_0 (G_{uu} u_1 + G_{uv} v_1^{(i)}) + \psi_0^{(i)} (G_{uv} u_1 + G_{vv} v_1^{(i)}) \right) \\
&\quad - \lambda_2 T e^{-\lambda_1 T} \left(G_u \phi_0 + G_v \psi_0^{(i)} \right), \quad \rho = 1.
\end{aligned} \tag{3.11}$$

In the inner region near the compartments we assume radial symmetry so that $\Delta_y \equiv \partial_\rho^2 + (2/\rho)\partial_\rho$, which is the spherically symmetric Laplacian. Here ρ is the radial variable defined as $\rho = |y|$. Note that $\hat{n} \cdot \nabla_y = \partial_{ny} \equiv -\partial_\rho$ because the normal vector, $\hat{n}$, points inward to the spherical compartments.

Throughout this section, the partial derivatives of the functions F and G are evaluated at the leading order steady state solutions. The leading order steady solution for $u(x,t)$ is u_0 in its outer region and $u_0 + c_1/\rho$ in its inner region, $|x - x_1| \sim \varepsilon$. The leading order steady solution for $v(x,t)$ is v_0 in its outer region and $v_0 + c_2/\rho$ in its inner region, $|x - x_2| \sim \varepsilon$. On the boundaries of the compartments we have that $\rho = 1$. We use the following notation for the partial derivatives of the functions F and G when being evaluated at the leading order steady state solutions. We write F_u for $F_u(u_0 + c_1, v_0)$ and F_v for $F_v(u_0 + c_1, v_0)$. Similarly we write G_u for $G_u(u_0, v_0 + c_2)$ and G_v for $G_v(u_0, v_0 + c_2)$. Recall that u_0, v_0, c_1, and c_2 are determined from the equations in (3.3).

Throughout the analysis in this section we will be using the method of matched asymptotic expansions as we did in Chapter 2. The following matching conditions must hold to all orders in ε,

$$\begin{aligned} \phi_0 + \varepsilon\phi_1 + \varepsilon^2\phi_2 + \ldots &= \phi_0^{(i)} + \varepsilon\phi_1^{(i)} + \varepsilon^2\phi_2^{(i)} + \ldots \\ \psi_0 + \varepsilon\psi_1 + \varepsilon^2\psi_2 + \ldots &= \psi_0^{(i)} + \varepsilon\psi_1^{(i)} + \varepsilon^2\psi_2^{(i)} + \ldots . \end{aligned} \tag{3.12}$$

From the equations for ϕ_0 and ψ_0 in (3.8) we have that ϕ_0 and ψ_0 are constants. For the inner solutions we solve the equations for $\phi_0^{(i)}$ and $\psi_0^{(i)}$ from (3.11). The inner functions $\phi_0^{(i)}$ and $\psi_0^{(i)}$ are spherically symmetric. From the matching condition we have that $\phi_0^{(i)} \to \phi_0$ and $\psi_0^{(i)} \to \psi_0$ as $\rho \to \infty$.

$$\begin{aligned} \phi_0^{(i)} &= \phi_0 + \frac{e_1}{\rho}, \\ \psi_0^{(i)} &= \psi_0 + \frac{e_2}{\rho}. \end{aligned} \tag{3.13}$$

The boundary conditions for $\phi_0^{(i)}$ and $\psi_0^{(i)}$ give

$$\begin{aligned} e_1 &= e^{-\lambda_1 T}\left(F_u(\phi_0 + e_1) + F_v\psi_0\right) \\ e_2 &= e^{-\lambda_1 T}\left(G_u\phi_0 + G_v(\psi_0 + e_2)\right). \end{aligned} \tag{3.14}$$

Next we match the outer and inner solutions using (3.12)

$$\begin{aligned}\phi_0 + \varepsilon\phi_1 + \ldots &= \phi_0 + \frac{e_1\varepsilon}{|x-x_1|} + \ldots \\ \psi_0 + \varepsilon\psi_1 + \ldots &= \psi_0 + \frac{e_2\varepsilon}{|x-x_2|} + \ldots ,\end{aligned}$$

which gives

$$\begin{aligned}\phi_1(x) &\sim \frac{e_1}{|x-x_1|}, \quad x \to x_1 \\ \psi_1(x) &\sim \frac{e_2}{|x-x_2|}, \quad x \to x_2.\end{aligned} \tag{3.15}$$

From (3.15) we see that both ϕ_1 and ψ_1 are proportional to the free-space Green's function $G_F(x) = \frac{1}{4\pi|x-x_1|}$ of the Laplace equation, which satisfies $\Delta G_F = -\delta(x-x_1)$. Therefore we add in a δ-term of strength $4\pi e_1$ at the point x_1 and a δ-term of strength $4\pi e_2$ at x_2. This extends the domain of the outer BVPs for ϕ_1 and ψ_1 in (3.8) to all of Ω_1. Therefore ϕ_1 and ψ_1 are now determined by the following BVPs,

$$\begin{aligned}\Delta_x\phi_1 - (\alpha_1^2 + \tau_u\lambda_1)\phi_0 &= -4\pi e_1\delta(x-x_1), \quad x \in \Omega_1 \\ \Delta_x\psi_1 - (\alpha_2^2 + \tau_v\lambda_1)\psi_0 &= -4\pi e_2\delta(x-x_2), \quad x \in \Omega_1.\end{aligned} \tag{3.16}$$

Integrating the equations in (3.16) over the domain Ω_1 and applying the Divergence theorem yields the following solvability condition,

$$\begin{aligned}e_1 &= \left(\frac{\alpha_1^2 + \tau_u\lambda_1}{3}\right)\phi_0 \\ e_2 &= \left(\frac{\alpha_2^2 + \tau_v\lambda_1}{3}\right)\psi_0.\end{aligned} \tag{3.17}$$

Combining (3.14) and (3.17) we obtain

$$\begin{pmatrix} (e^{\lambda_1 T} - F_u)(\alpha_1^2 + \tau_u\lambda_1) - 3F_u & -3F_v \\ -3G_u & (e^{\lambda_1 T} - G_v)(\alpha_2^2 + \tau_v\lambda_1) - 3G_v \end{pmatrix} \begin{pmatrix}\phi_0 \\ \psi_0\end{pmatrix} = \begin{pmatrix}0 \\ 0\end{pmatrix}. \tag{3.18}$$

The eigenvalue problem in (3.18) is a nonlinear eigenvalue problem. This is because of the exponential term $e^{\lambda_1 T}$. Setting $T = 0$ leads to the leading order eigenvalue problem derived in (2.35) for the model without delay. We will use the matrix eigenvalue problem in (3.18) more later. It is the main result of this section.

Now we derive equations which can be solved to determine the correction term λ_2. Solving for ϕ_1 and ψ_1 in (3.16) leads to

$$\begin{aligned}\phi_1(x) &= 4\pi e_1 G_n(x; x_1) + \chi_3 \\ \psi_1(x) &= 4\pi e_2 G_n(x; x_2) + \chi_4,\end{aligned} \tag{3.19}$$

where χ_3 and χ_4 are constants to to be determined. The solutions in (3.19) are written in terms of the Neumann Green's function. This was defined in Chapter 2 but we rewrite it here for easier reference. The Neumann Green's function $G_n(x;x_0)$ satisfies

$$\begin{aligned}
\Delta_x G_n(x;x_0) &= 1/|\Omega_1| - \delta(x-x_0), \quad x \in \Omega_1 \\
\partial_{nx} G_n(x;x_0) &= 0, \quad x \in \partial\Omega_1 \\
\int_{\Omega_1} G_n(x;x_0)\, dV &= 0,
\end{aligned} \tag{3.20}$$

where $|\Omega_1| = \frac{4\pi}{3}$, the volume of the unit sphere. For a sphere with radius R we can write $G_n(x;x_0)$ explicitly [6] as

$$G_n(x;x_0) = \frac{1}{4\pi|x-x_0|} + R_n(x;x_0), \tag{3.21}$$

where $R_n(x;x_0)$ is the regular part of this Green's function given by

$$\begin{aligned}
R_n(x;x_0) =& \frac{R}{4\pi} \frac{1}{r\left|\frac{R^2}{r^2}x - x_0\right|} + \frac{1}{8\pi R^3}(r^2 + r_0^2) - \frac{7}{10\pi R} \\
&+ \frac{1}{4\pi R} \log\left(\frac{2R^2}{R^2 - r r_0 \cos\gamma + r\left|\frac{R^2}{r^2}x - x_0\right|} \right).
\end{aligned} \tag{3.22}$$

In (3.22), $\cos\gamma = \cos\theta\cos\theta_0 + \sin\theta\sin\theta_0\cos(\phi-\phi_0)$ and (r_0,θ_0,ϕ_0) denotes the spherical coordinates of x_0.

Now we consider the inner problems for $\phi_1^{(i)}$ and $\psi_1^{(i)}$ from (3.11). We begin by using (3.12) to determine the behavior of both these functions as $\rho \to \infty$,

$$\begin{aligned}
\phi_0 + \varepsilon\left(\frac{e_1}{\varepsilon\rho} + 4\pi e_1 R_n(x_1;x_1) + \chi_3\right) + \varepsilon^2\phi_2 + \ldots &= \phi_0 + \frac{e_1}{\rho} + \varepsilon\phi_1^{(i)} \\
\psi_0 + \varepsilon\left(\frac{e_2}{\varepsilon\rho} + 4\pi e_2 R_n(x_2;x_2) + \chi_4\right) + \varepsilon^2\psi_2 + \ldots &= \psi_0 + \frac{e_2}{\rho} + \varepsilon\psi_1^{(i)}.
\end{aligned} \tag{3.23}$$

The matching condition in (3.23) gives

$$\begin{aligned}
\phi_1^{(i)} &\to 4\pi e_1 R_n(x_1;x_1) + \chi_3, \quad \rho \to \infty \\
\psi_1^{(i)} &\to 4\pi e_2 R_n(x_2;x_2) + \chi_4, \quad \rho \to \infty.
\end{aligned}$$

Using this condition as well as the fact that both $\phi_1^{(i)}$ and $\psi_1^{(i)}$ satisfy the spherically symmetric Laplace equation in (3.11), we can solve for $\phi_1^{(i)}$ and $\psi_1^{(i)}$ to obtain

$$\begin{aligned}
\phi_1^{(i)} &= 4\pi e_1 R_n(x_1;x_1) + \chi_3 + \frac{A_3}{\rho} \\
\psi_1^{(i)} &= 4\pi e_2 R_n(x_2;x_2) + \chi_4 + \frac{A_4}{\rho}.
\end{aligned} \tag{3.24}$$

We will determine A_3 and A_4 from the boundary conditions for $\phi_1^{(i)}$ and $\psi_1^{(i)}$ in (3.11) shortly. First we use the matching condition to determine the behavior of ϕ_2 and ψ_2 near their respective compartments. We then use this to extend the domain for the BVPs for ϕ_2 and ψ_2. The matching condition from (3.12) was rewritten in (3.23). If we substitute $\phi_1^{(i)}$ and $\psi_1^{(i)}$ from (3.24) into (3.23) we obtain the following result after matching,

$$\phi_2 \sim \frac{A_3}{|x-x_1|}, \quad x \to x_1$$
$$\psi_2 \sim \frac{A_4}{|x-x_2|}, \quad x \to x_2.$$

Therefore we can consider the two points x_1 and x_2 as point sources and add in delta functions at x_1 and x_2. This extend the domain of the BVPs for ϕ_2 and ψ_2 from (3.8) to all of Ω_1. The new BVPs are

$$\begin{aligned}
&\Delta_x\phi_2 = (\alpha_1^2+\tau_u\lambda_1)\phi_1 + \tau_u\lambda_2\phi_0 - 4\pi A_3\delta(x-x_1), \quad x\in\Omega_1, \qquad \partial_{nx}\phi_2 = 0, x\in\partial\Omega_1\\
&\Delta_x\psi_2 = (\alpha_2^2+\tau_v\lambda_1)\psi_1 + \tau_v\lambda_2\psi_0 - 4\pi A_4\delta(x-x_2), \quad x\in\Omega_1, \qquad \partial_{nx}\psi_2 = 0, x\in\partial\Omega_1.
\end{aligned} \tag{3.25}$$

Integrating these BVPs over Ω_1, applying the Divergence theorem, and using the integral condition from (3.20) leads to

$$\begin{aligned}
A_3 &= \frac{(\alpha_1^2+\tau_u\lambda_1)\chi_3 + \tau_u\lambda_2\phi_0}{3}\\
A_4 &= \frac{(\alpha_2^2+\tau_v\lambda_1)\chi_4 + \tau_v\lambda_2\psi_0}{3}.
\end{aligned} \tag{3.26}$$

The boundary conditions for $\phi_1^{(i)}$ and $\psi_1^{(i)}$ from (3.11) give us two other equations for A_3 and A_4. Equating these equations for A_3 and A_4 with the equations for A_3 and A_4 from (3.26) yields the following after rearranging,

$$\begin{aligned}
&\begin{pmatrix} (e^{\lambda_1 T}-F_u)(\alpha_1^2+\tau_u\lambda_1) - 3F_u & -3F_v \\ -3G_u & (e^{\lambda_1 T}-G_v)(\alpha_2^2+\tau_v\lambda_1) - 3G_v \end{pmatrix}\begin{pmatrix}\chi_3\\ \chi_4\end{pmatrix} +\\
&\begin{pmatrix} e^{\lambda_1 T}-F_u & 0 \\ 0 & e^{\lambda_1 T}-G_v \end{pmatrix}\begin{pmatrix}\tau_u\lambda_2\phi_0\\ \tau_v\lambda_2\psi_0\end{pmatrix} = \begin{pmatrix} 3F_u 4\pi R_n(x_1;x_1) & 3F_v 4\pi G_n(x_1;x_2) \\ 3G_u 4\pi G_n(x_2;x_1) & 3G_v 4\pi R_n(x_2;x_2)\end{pmatrix}\begin{pmatrix}e_1\\ e_2\end{pmatrix}\\
&+\begin{pmatrix} 3\phi_0^{(i)}(F_{uu}u_1^{(i)}+F_{uv}v_1) + 3\psi_0(F_{uv}u_1^{(i)}+F_{vv}v_1) - 3\lambda_2 T\left(F_u\phi_0^{(i)}+F_v\psi_0\right) \\ 3\phi_0(G_{uu}u_1+G_{uv}v_1^{(i)}) + 3\psi_0^{(i)}(G_{uv}u_1+G_{vv}v_1^{(i)}) - 3\lambda_2 T\left(G_u\phi_0+G_v\psi_0^{(i)}\right)\end{pmatrix}
\end{aligned} \tag{3.27}$$

The first equation in (3.27) is evaluated on the boundary of the first compartment located at x_1. Therefore $\rho = 1$ which implies $|x - x_1| = \varepsilon$. For the second equation the terms are evaluated on the boundary of the second compartment located at x_2. Therefore $\rho = 1$ so that $|x - x_2| = \varepsilon$. For example, the term $\phi_0^{(i)} = \phi_0 + e_1/\rho$ is simplified to $\phi_0 + e_1$.

The system in (3.27) can be written in the form

$$\boldsymbol{A}\begin{pmatrix}\chi_3\\ \chi_4\end{pmatrix} = \boldsymbol{N}_{\lambda_2},$$

where $\boldsymbol{A}$ is the matrix multiplying the vector $(\chi_1, \chi_2)^\top$ and $\boldsymbol{N}_{\lambda_2}$ is the vector containing all the other terms from (3.27). Note that $\det(\boldsymbol{A}) = 0$ because λ_1 is an eigenvalue of $\boldsymbol{A}$. To solve the above system for λ_2 we use a solvability condition which is basically just an application of the Fredholm alternative. If we denote $\boldsymbol{A}^*$ as the Hermitian/conjugate transpose of $\boldsymbol{A}$ and let $\boldsymbol{Y}$ be a non zero vector such that $\boldsymbol{A}^*\boldsymbol{Y} = 0$ then it follows that

$$\left\langle \boldsymbol{A}\begin{pmatrix}\chi_3\\ \chi_4\end{pmatrix}, \boldsymbol{Y} \right\rangle = \langle \boldsymbol{N}_{\lambda_2}, \boldsymbol{Y}\rangle$$

$$\left\langle \begin{pmatrix}\chi_3\\ \chi_4\end{pmatrix}, \boldsymbol{A}^*\boldsymbol{Y} \right\rangle = \langle \boldsymbol{N}_{\lambda_2}, \boldsymbol{Y}\rangle$$

$$0 = \langle \boldsymbol{N}_{\lambda_2}, \boldsymbol{Y}\rangle .$$

The angled brackets $\langle .\ ,\ .\rangle$ denote the standard complex inner product. Provided that we can find a nontrivial vector $\boldsymbol{Y}$ in the kernel of $\boldsymbol{A}^*$, then we can solve the last equation above for λ_2. Knowing the value of λ_2 makes the approximation to λ more accurate.

The reason for going to a higher order in ε to find λ_2 in this section is to show how the method of matched asymptotics works when applied to the model with delay in (3.2). It also shows how the analysis is identical to the stability analysis in §2.3. Setting the delay $T = 0$ results in the same results from §2.3. In other sections of this chapter we will do similar calculations without showing as many details. Although we have obtained the correction term λ_2, we do not use its value in this chapter. In Chapter 2, when there was no delay, we could approximate the eigenvalues of the PDE eigenvalue problem (2.28) numerically in Comsol. Since there was no delay the

PDE eigenvalue problem was linear. We could then compare the numerical solution of λ with the asymptotic result $\lambda_1\varepsilon + \lambda_2\varepsilon^2$ as we did in Table 2.1. In this chapter the delay leads to the nonlinear eigenvalue problem in (3.4). We have tried using Comsol to solve (3.4) for λ but have not had any success getting consistent values.

We are more concerned with the dynamics of (3.2) and the leading order part of the eigenvalue λ_1 is more useful in this regard. In the examples later in this chapter we use the eigenvalue problem in (3.18) to solve for λ_1. To leading order, the $\mathrm{Re}(\lambda_1)$ determines the stability of the steady state solutions. Knowing this helps determine the type of dynamics that can occur for the model with large delay. For instance, we can find cases for which $\lambda_1 = \pm\omega i$ and show that a Hopf bifurcation occurs. Using (3.18) we can find critical delay values which result in the solutions of (3.2) having sustained oscillations. An example of this will be shown later in §3.5.2.

Although we will not be using them explicitly, for completeness we note that the the eigenfunctions of (3.4) have the form

$$\phi = \phi_0 + \varepsilon\phi_1 + \ldots, \quad \phi_1 = \frac{e_1}{|x - x_1|} + 4\pi e_1 R_n(x; x_1) + \chi_3$$
$$\psi = \psi_0 + \varepsilon\psi_1 + \ldots, \quad \psi_1 = \frac{e_2}{|x - x_2|} + 4\pi e_2 R_n(x; x_2) + \chi_4,$$

Smaller Delay, $s = O(1)$

Here we assume that the delay is O(1). We let $s = T$ where $T = O(1)$. The eigenvalue λ is still $O(\varepsilon)$. The exponential term in the inner boundary conditions from (3.4) becomes

$$e^{-\lambda T} = e^{-(\varepsilon\lambda_1 + \varepsilon^2\lambda_2)T} = 1 - \varepsilon\lambda_1 T + O(\varepsilon^2).$$

Therefore the delay term T does not show up in the leading order eigenvalue problem. The leading order term of the eigenvalue, λ_1, is determined by the eigenvalue problem in (2.35) with no delay. Therefore we expect to see similar qualitative dynamics between the model without delay and the model with an O(1) delay. This is a result of the chosen scaling of the decay parameters in (3.2). We note that in this case we can continue the analysis and solve for λ_2 but it does not provide anything interesting. One may think it is possible to get some interesting behavior if $\lambda_1 = 0$ and $\lambda_2 = \pm\omega i$. This is not possible and can be seen if the analysis is continued. Therefore it is not

possible to get a Hopf bifurcation using the delay as the bifurcation parameter when $s = O(1)$.

3.3 Time Dependent Approximations of the PDE Model

In this section we find an approximate time dependent solution to (3.2) when $s = O(1/\varepsilon)$. The main result of this section is the reduction of (3.2) to a system of DDAEs. The time dependent asymptotic approximation derived in this section is valid for times when the solution is not changing too quickly or when the solution is near equilibrium. It turns out that the approximation is fairly good for all time except for any initial transients where the solution changes rapidly. We did the exact same calculation in §2.4 when there was no delay. The difference in this section is that we obtain a system of DDAEs as opposed to a system of DAEs. Moreover, the DAEs in §2.4 could be converted to a system of ODEs after solving the implicit algebraic equations for c_1 and c_2. This is not the case in this section because the addition of delay complicates the implicit relationships and they can not be solved in general. In general, the DDAEs derived here have more complicated dynamics and also present more challenges when finding numerical approximations to their solutions.

From the stability analysis in §3.2 we know that the delay s has to be of $O(1/\varepsilon)$ to have a significant affect on the dynamics of the model (3.2). Therefore we assume here that $s = T/\varepsilon$ where $T = O(1)$. We begin by introducing the slow time variable $\tau = \varepsilon t$ into (3.2). Therefore we let $u(t) = u(\frac{1}{\varepsilon}\tau) \equiv \widetilde{u}(\tau)$. For notational convenience we will drop the $\sim$ and just write $u(\tau)$. The system in (3.2) becomes

$$\begin{aligned}
\varepsilon\tau_u \frac{\partial u}{\partial \tau} &= \Delta_x u - \alpha_1^2 \varepsilon u, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} u &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon \partial_{nx} u &= F(u(x, \tau - T), v(x, \tau - T)), \quad x \in \partial\Omega_{\varepsilon_1} \\
\\
\varepsilon\tau_v \frac{\partial v}{\partial \tau} &= \Delta_x v - \alpha_2^2 \varepsilon v, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} v &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon \partial_{nx} v &= G(u(x, \tau - T), v(x, \tau - T)), \quad x \in \partial\Omega_{\varepsilon_2}.
\end{aligned} \tag{3.28}$$

We will ignore the history functions u_h and v_h for now and discuss them later. We will find an asymptotic approximation to the time dependent problem in (3.28) using the method of matched asymptotics. The analysis is identical to §2.4 so we will omit some of the details.

First we expand the outer and inner solutions in terms of ε

$$\begin{aligned} u(x,\tau) &= u_0(x,\tau) + \varepsilon u_1(x,\tau) + \dots \\ v(x,\tau) &= v_0(x,\tau) + \varepsilon v_1(x,\tau) + \dots \\ u^{(i)}(y,\tau) &= u_0^{(i)}(y,\tau) + \varepsilon u_1^{(i)}(y,\tau) + \dots \\ v^{(i)}(y,\tau) &= v_0^{(i)}(y,\tau) + \varepsilon v_1^{(i)}(y,\tau) + \dots, \end{aligned} \tag{3.29}$$

and substitute these into (3.28). Equating powers of ε we obtain the following outer problems

$$\begin{aligned} \Delta_x u_0 &= 0, \quad x \in \Omega_1 \backslash \{x_1\}, & \partial_{nx} u_0 &= 0, \quad x \in \partial\Omega_1 \\ \tau_u \frac{\partial u_0}{\partial \tau} &= \Delta_x u_1 - \alpha_1^2\, u_0, \quad x \in \Omega_1 \backslash \{x_1\}, & \partial_{nx} u_1 &= 0, \quad x \in \partial\Omega_1 \\ \tau_u \frac{\partial u_1}{\partial \tau} &= \Delta_x u_2 - \alpha_1^2 u_1, \quad x \in \Omega_1 \backslash \{x_1\}, & \partial_{nx} u_2 &= 0, \quad x \in \partial\Omega_1 \end{aligned} \tag{3.30}$$

$$\begin{aligned} \Delta_x v_0 &= 0, \quad x \in \Omega_1 \backslash \{x_2\}, & \partial_{nx} v_0 &= 0, \quad x \in \partial\Omega_1 \\ \tau_v \frac{\partial v_0}{\partial \tau} &= \Delta_x v_1 - \alpha_2^2\, v_0, \quad x \in \Omega_1 \backslash \{x_2\}, & \partial_{nx} v_1 &= 0, \quad x \in \partial\Omega_1 \\ \tau_v \frac{\partial v_1}{\partial \tau} &= \Delta_x v_2 - \alpha_2^2 v_1, \quad x \in \Omega_1 \backslash \{x_2\}, & \partial_{nx} v_2 &= 0, \quad x \in \partial\Omega_1. \end{aligned}$$

Near the compartments, in the inner regions, we rescale space using $\rho = |y| = \frac{|x - x_j|}{\varepsilon}$ in (3.28) to obtain the following inner problems

$$\begin{aligned} 0 &= \Delta_y u_0^{(i)}, \quad \rho > 1 \\ -\partial_\rho u_0^{(i)} &= F(u_{0_T}^{(i)}, v_{0_T}), \quad \rho = 1 \\ 0 &= \Delta_y u_1^{(i)}, \quad \rho > 1 \\ -\partial_\rho u_1^{(i)} &= F_u(u_{0_T}^{(i)}, v_{0_T}) u_{1_T}^{(i)} + F_v(u_{0_T}^{(i)}, v_{0_T}) v_{1_T}, \quad \rho = 1 \\ 0 &= \Delta_y v_0^{(i)}, \quad \rho > 1 \\ -\partial_\rho v_0^{(i)} &= G(u_{0_T}, v_{0_T}^{(i)}), \quad \rho = 1 \\ 0 &= \Delta_y v_1^{(i)}, \quad \rho > 1 \\ -\partial_\rho v_1^{(i)} &= G_u(u_{0_T}, v_{0_T}^{(i)}) u_{1_T} + G_v(u_{0_T}, v_{0_T}^{(i)}) v_{1_T}^{(i)}, \quad \rho = 1. \end{aligned} \tag{3.31}$$

We will use the notation u_{0_T} to mean $u_0(\tau - T)$ and so on, throughout this section.

The matching condition for the outer and inner problems works the same as it did before in §2.4.

$$\begin{aligned} u_0(\tau) + \varepsilon u_1(x,\tau) + \varepsilon^2 u_2(x,\tau) + \ldots &= u_0^{(i)}(\rho,\tau) + \varepsilon u_1^{(i)}(\rho,\tau) + \varepsilon^2 u_2^{(i)}(\rho,\tau) + \ldots \\ v_0(\tau) + \varepsilon v_1(x,\tau) + \varepsilon^2 v_2(x,\tau) + \ldots &= v_0^{(i)}(\rho,\tau) + \varepsilon v_1^{(i)}(\rho,\tau) + \varepsilon^2 v_2^{(i)}(\rho,\tau) + \ldots . \end{aligned} \tag{3.32}$$

From the outer problems for u_0 and v_0 in (3.30) we have u_0 and v_0 are functions of τ only. These functions are constant in space. The matching condition in (3.32) implies that that $u_0^{(i)} \to u_0(\tau)$ and $v_0^{(i)} \to v_0(\tau)$ as $\rho \to \infty$. This along with the BVPs for $u_0^{(i)}$ and $v_0^{(i)}$ in (3.31) yields

$$\begin{aligned} u_0^{(i)}(y,\tau) &= u_0(\tau) + \frac{c_1(\tau)}{\rho}, \quad c_1(\tau) = F(u_{0_T} + c_{1_T}, v_{0_T}) \\ v_0^{(i)}(y,\tau) &= v_0(\tau) + \frac{c_2(\tau)}{\rho}, \quad c_2(\tau) = G(u_{0_T}, v_{0_T} + c_{2_T}) \quad . \end{aligned}$$

Next, we use the matching condition in (3.32) to get the behavior for u_1 as $x \to x_1$ and v_1 as $x \to x_2$.

$$\begin{aligned} u_1(x,\tau) &\sim \frac{c_1(\tau)}{|x - x_1|}, \quad x \to x_1 \\ v_1(x,\tau) &\sim \frac{c_2(\tau)}{|x - x_2|}, \quad x \to x_2. \end{aligned}$$

As in §2.4, we can extend the domain from $\Omega_1 \backslash \{x_1\}$ and $\Omega_1 \backslash \{x_2\}$ to all of Ω_1 by including δ terms of appropriate strengths at the singularity points x_1 and x_2. Doing this for the BVPs for u_1 and v_1 from (3.30) leads to

$$\begin{aligned} \tau_u \frac{\partial u_0}{\partial \tau} &= \Delta_x u_1 - \alpha_1^2 u_0 + 4\pi c_1 \delta(x - x_1), \quad x \in \Omega_1, \qquad \partial_{nx} u_1 = 0, \quad x \in \partial\Omega_1 \\ \tau_v \frac{\partial v_0}{\partial \tau} &= \Delta_x v_1 - \alpha_2^2 v_0 + 4\pi c_2 \delta(x - x_2), \quad x \in \Omega_1, \qquad \partial_{nx} v_1 = 0, \quad x \in \partial\Omega_1. \end{aligned} \tag{3.33}$$

Integrating the above equations over the domain Ω_1 and using the Divergence theorem leads to the following DDAEs for $u_0(\tau)$, $v_0(\tau)$, $c_1(\tau)$, and $c_2(\tau)$

$$\begin{aligned} \begin{pmatrix} \frac{\tau_u}{3} u_0 \\ \frac{\tau_v}{3} v_0 \end{pmatrix}' &= - \begin{pmatrix} \frac{\alpha_1^2}{3} & 0 \\ 0 & \frac{\alpha_2^2}{3} \end{pmatrix} \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} + \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \\ \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} &= \begin{pmatrix} F(u_{0_T} + c_{1_T}, v_{0_T}) \\ G(u_{0_T}, v_{0_T} + c_{2_T}) \end{pmatrix} \end{aligned} \tag{3.34}$$

In §2.4 this same analysis was done on (3.1) which resulted in a DAE system for u_0, v_0, c_1, and c_2 which can be found in (2.47). This system from (2.47) can be obtained by setting $T = 0$ in (3.34). For the examples of the model with no delay considered in §2.5, the implicit relationships for c_1 and c_2 could be solved for in terms of u_0 and v_0. This led to a system of two ODEs for u_0 and v_0.

The main challenge presented with (3.34) is that $c_1(\tau)$ and $c_2(\tau)$ are defined implicitly in terms of their past history values. There is no way in general of solving for c_1 or c_2 as functions of u_0 and v_0. Only in the case that $F_u = G_v = 0$ can this be done. If $F_u = G_v = 0$ then solving (3.34) is relatively straight forward using a regular DDE solver such as **dde23** in Matlab. We will discuss how we have chosen to solve (3.34) numerically in §3.4.

Continuing with the method of matched asymptotics, we write the solutions to (3.33) using the Green's function defined in (3.21),

$$\begin{aligned} u_1(x,\tau) &= 4\pi c_1(\tau) G_n(x;x_1) + \chi_1(\tau) \\ v_1(x,\tau) &= 4\pi c_2(\tau) G_n(x;x_2) + \chi_2(\tau). \end{aligned}$$

The functions $\chi_1(\tau)$ and $\chi_2(\tau)$ are found by solving a system of DDAEs which we derive shortly.

At the next order in the inner solution, we solve for $u_1^{(i)}(y,\tau)$ and $v_1^{(i)}(y,\tau)$. From matching we can determine their behavior away from the compartments as $\rho \to \infty$. This along with the BVPs for $u_1^{(i)}(y,\tau)$ and $v_1^{(i)}(y,\tau)$ in (3.31) gives

$$\begin{aligned} u_1^{(i)} &= 4\pi c_1(\tau) R_n(x_1;x_1) + \chi_1(\tau) + \frac{A_1(\tau)}{\rho} \\ v_1^{(i)} &= 4\pi c_2(\tau) R_n(x_2;x_2) + \chi_2(\tau) + \frac{A_2(\tau)}{\rho}. \end{aligned} \tag{3.35}$$

Here the function $R_n(x;x_0)$ is the regular part of the Green's function defined in (3.22). The functions $A_1(\tau)$ and $A_2(\tau)$ are determined by the boundary conditions for $u_1^{(i)}(y,\tau)$ and $v_1^{(i)}(y,\tau)$ in (3.31).

$$\begin{aligned} A_1(\tau) &= F_{u_T}(4\pi c_{1_T} R_n(x_1;x_1) + \chi_{1_T} + A_{1_T}) + F_{v_T} v_1(x_1, \tau - T) \\ A_2(\tau) &= G_{u_T} u_1(x_2, \tau - T) + G_{v_T}(4\pi c_{2_T} R_n(x_2;x_2) + \chi_{2_T} + A_{2_T}). \end{aligned} \tag{3.36}$$

Every time dependent function on the right hand side of (3.36) is delayed. We are using the notation $v_1(x_1,\tau)$ and $u_1(x_2,\tau)$ to mean $v_1(x_1,\tau) \equiv v_1\big|_{|x-x_1|=\varepsilon}$ and

$u_1(x_2,\tau) \equiv u_1\big|_{|x-x_2|=\varepsilon}$. The partial derivatives are evaluated at the leading order solutions. We write F_{u_T} for $F_u(u_{0_T}+c_{1_T}, v_{0_T})$ and F_{v_T} for $F_v(u_{0_T}+c_{1_T}, v_{0_T})$. Similarly we write G_{u_T} for $G_u(u_{0_T}, v_{0_T}+c_{2_T})$ and G_{v_T} for $G_v(u_{0_T}, v_{0_T}+c_{2_T})$.

Next, we use the matching condition in (3.32) to determine the behavior of u_2 as $x \to x_1$ and v_2 $x \to x_2$. As we have done several times now, we add in delta functions to the BVPs for u_2 and v_2 from (3.30) and extend the domain of these BVPs to all of Ω_1,

$$\begin{aligned}
\tau_u \frac{\partial u_1}{\partial \tau} &= \Delta_x u_2 - \alpha_1^2 u_1 + 4\pi A_1 \delta(x-x_1), \quad x \in \Omega_1 \backslash \{x_1\}, \quad \partial_{nx} u_2 = 0, \quad x \in \partial\Omega_1 \\
\tau_v \frac{\partial v_1}{\partial \tau} &= \Delta_x v_2 - \alpha_2^2 v_1 + 4\pi A_2 \delta(x-x_2), \quad x \in \Omega_1 \backslash \{x_2\}, \quad \partial_{nx} v_2 = 0, \quad x \in \partial\Omega_1.
\end{aligned}$$

Integrating the above equations and using the Divergence Theorem leads to two differential equations for χ_1 and χ_2. Combing these two equations with the algebraic equations for A_1 and A_2 from (3.36) leads to the following DDAEs for $\chi_1(\tau)$, $\chi_2(\tau)$, $A_1(\tau)$, and $A_2(\tau)$.

$$\begin{aligned}
\begin{pmatrix} \frac{\tau_u}{3}\chi_1 \\ \frac{\tau_v}{3}\chi_2 \end{pmatrix}' &= -\begin{pmatrix} \frac{\alpha_1^2}{3} & 0 \\ 0 & \frac{\alpha_2^2}{3} \end{pmatrix} \begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} + \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \\
\begin{pmatrix} A_1 \\ A_2 \end{pmatrix} &= \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}_T + \begin{pmatrix} F_u & F_v \\ G_u & G_v \end{pmatrix}_T \begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix}_T + \begin{pmatrix} F_u & 0 \\ 0 & G_v \end{pmatrix}_T \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}_T,
\end{aligned} \tag{3.37}$$

The functions B_1 and B_2 are defined as

$$\boldsymbol{B} = \begin{pmatrix} B_1(\tau) \\ B_2(\tau) \end{pmatrix} = \begin{pmatrix} F_u 4\pi c_1 R_n(x_1;x_1) + F_v 4\pi c_2 G_n(x_1;x_2) \\ G_u 4\pi c_1 G_n(x_2;x_1) + G_v 4\pi c_2 R_n(x_2;x_2) \end{pmatrix}. \tag{3.38}$$

The same ODEs for χ_1 and χ_2 were derived in (2.49). Similar looking equations for A_1 and A_2 were also derived in (2.48). The main difference here is that everything on the right hand side of the equations for A_1 and A_2 in (3.37) is delayed. The equations for A_1 and A_2 in (2.48) had no delay and could be solved for in terms of χ_1 and χ_2. This is not the case in (3.37) because of the delay.

In this section we have approximated the PDE system in (3.2) by the systems of DDAEs found in (3.34) and (3.37). We will discuss how to approximate the solutions of these two systems numerically in §3.4. Upon solving the system in (3.34) for u_0, v_0, c_1, and c_2 and the system in (3.37) for χ_1, χ_2, A_1, and A_2, we obtain the time

dependent approximation of (3.28) as

$$\begin{aligned} u(x,\tau) &= u_0(\tau) + \varepsilon\left(\frac{c_1(\tau)}{|x-x_1|} + 4\pi c_1(\tau)R_n(x;x_1) + \chi_1(\tau)\right) \\ v(x,\tau) &= v_0(\tau) + \varepsilon\left(\frac{c_2(\tau)}{|x-x_2|} + 4\pi c_2(\tau)R_n(x;x_2) + \chi_2(\tau)\right). \end{aligned} \tag{3.39}$$

Then by setting $\tau = \varepsilon t$ we have approximate solutions to (3.2) for $s = T/\varepsilon$. In §3.5 we compare these approximations with the numerical approximations of the full system (3.2).

3.4 Numerical solutions of PDEs and DDAEs

In this section we discuss the numerical computations for solving the PDE system (3.2). We emphasize that these numerical computations are very different from the numerical computations for delayed ODEs since we are dealing with PDEs and delay in 3D. Moreover, because of the complicated geometry we must use a FEM software package. We have chosen to use Comsol. To our knowledge, we do not know of any FEM software which solves delay PDEs defined over complicated geometries. We will also discuss, in §3.4.3, the numerical solution of the DDAES defined in §3.3.

3.4.1 Method of Steps in Comsol

In this section we explain how we have used the Comsol and Matlab scripting languages to approximate solutions of (3.2) numerically using the method of steps [8]. The method of steps turns (3.2) into a series of time dependent problems without delay. Our method relies on the fact that we only require the history of the solution on the boundary of the compartments. The method we implement here could possibly be generalized to solve other PDEs with constant delays defined over complicated spatial domains.

First we consider (3.2) on the time interval $[0, s]$. On this interval, the functions $F(u(x,t-s), v(x,t-s))$ and $G(u(x,t-s), v(x,t-s))$ become $F(u_h, v_h)$ and $G(u_h, v_h)$. Since the functions u_h and v_h are the known history data, for $t \in [0, s]$ the system in (3.2) can be reduced to a problem without delay. We can use the standard time dependent solvers in Comsol on the interval $[0, s]$.

Next, we solve (3.2) for $t \in [s, 2s]$. The delayed arguments in the functions F and G are now accessing solution data on the previous delay interval, $[0, s]$. We create continuous extensions of the solution data on $[0, s]$ using cubic spline interpolation. Then on the interval $[s, 2s]$ we can replace the delayed arguments with known interpolant evaluations. Then we can use Comsol to solve (3.2) on $[s, 2s]$ again using its standard solvers. In general we first obtain the solution on the interval $[js, (j+1)s]$. Then this is used as history to solve the PDEs on the next interval $[(j+1)s, (j+2)s]$. In each step we are solving a PDE whose boundary conditions are accessing time and space dependent functions which were found on the previous delay interval. Therefore we can use the standard time integrators in Comsol. We choose to use the same BDF methods as used in Comsol for Chapter 2.

There are several remarks that need to be made when implementing the method of steps in Comsol. First, the solutions of (3.2) will usually have discontinuities in their first derivatives at $t = 0$. This is because the history data will not in general satisfy the PDEs. Because of the delay, the discontinuity in the first derivative at $t = 0$ will propagate to a discontinuity in the second derivative at $t = s$ which then propagates to a discontinuity in the third derivative at $t = 2s$ and so on. Care must be taken when stepping through each of these discontinuities.

For example, if the discontinuities are ignored, the time stepper in Comsol will try to step through the discontinuities using its error control algorithm. The time stepping algorithm will choose the largest time step possible while still satisfying the given error tolerance. Upon reaching a point where the solution has a derivative discontinuity, the solver will take smaller and smaller time steps as to meet the required error tolerance. The time steps become smaller to counter the effect of the derivative discontinuity. The error estimate will be higher because of the derivative discontinuity. This leads to a dramatic decrease in step size. This is not efficient.

We force Comsol to step to the point in time where the derivative discontinuity is. These points are easily found because the discontinuities occur at integer multiples of the delay. The default setting in Comsol is to give it a time interval on which to compute the solution, $[0, t_{\text{out}}]$. The solver will often take a time step past t_{out} and then interpolate back to output the solution at the point t_{out}. In our method of steps code the last time step t_{out} will always be a multiple of the delay s. Therefore

there is a derivative discontinuity at t_{out}. We want the solver to step exactly to the discontinuity at $t = t_{\text{out}}$ and include it as a step point. We can force this in Comsol by using the option for strict time stepping. When the strict time stepping option is turned on, the solver will step exactly to t_{out}. When using this option along with the method of steps, all the points of discontinuity will be included as step points.

After stepping to each of these discontinuities in time we force Comsol to do a cold restart. This means that the simulation is restarted and continued from the last time step. A first order method is used and a small initial time step, determined by the error estimate, is used too. As the time stepping progresses, the order of the method and the size of the time steps increase until the next point in time is reached which has a derivative discontinuity. We then stop the solver and do a cold restart. This is done for each multiple of the delay.

To evaluate the delayed arguments in the boundary conditions on the interval $[(j+1)s, (j+2)s]$, we need to store the previous discrete solution data which was obtained on the interval $[js, (j+1)s]$. Once the solution values are stored we need to create a continuous extension of it. This is done using interpolation. Since the delay is only in the boundary conditions, we only need to save history data on the boundaries of the compartments within the spatial domain. In theory we need to select a set of points on the surface of the compartments and save the solution values for u and v at those points. Since the compartments are small and act as point sources, the solutions are basically spherically symmetric near and on the compartments. If we assume the solutions are constant on the compartment surfaces, then the solution values are only needed at a single point on each compartment. Therefore we store $u(x,t)$ at one point on the surface of Ω_{ε_1} as well as one point on the surface of Ω_{ε_2} for every time step in $[0, t_{\text{out}}]$. We also store $v(x,t)$ at one point on the surface of Ω_{ε_1} as well as one point on the surface of Ω_{ε_2} for every time step in $[0, t_{\text{out}}]$. These solution values can be stored in four different vectors. One vector contains $u(t)$ at a point on Ω_{ε_1}, another vector contains $u(t)$ at a point on Ω_{ε_2}, another vector contains $v(t)$ at a point on Ω_{ε_1}, and the fourth vector contains $v(t)$ at a point on Ω_{ε_2}.

Then a continuous extension of this data, stored in the four vectors, is needed to approximate $u(x, t-s)$ and $v(x, t-s)$ for $t \in [(j+1)s, (j+2)s]$. This is accomplished by using a built in cubic spline interpolant function within Comsol. The interpolants

are only time dependent functions since the solution data are coming from single points in the spatial domain, but at multiple times. These evaluations of interpolants are passed to the delayed arguments of the functions F and G in the compartment boundary conditions in (3.2).

After solving the BVP on the interval $[js, (j+1)s]$, we need to move to the next interval $[(j+1)s, (j+2)s]$. When starting up the solver again for the next time step we need to use the correct initial condition. The initial condition is the solution that was output last on the previous interval at $t = (j+1)s$. This solution is defined on the entire 3D spatial domain. Comsol has an option to restart a time dependent computation and use the last output solution as an initial condition. The last output solution in this context is the one from the previous delay interval at $t = (j+1)s$. We use this option to start up the computation on the interval $[(j+1)s, (j+2)s]$.

In summary we solve (3.2) on the first delay interval $[0, s]$ and stop exactly at the end of the interval so as not to step beyond the first derivative discontinuity at $t = s$. The solution is then stored at all the step points on the interval which the solver stepped through. We use the spherical symmetry to our advantage around the compartments and store the solution values at just a single point on each compartment. These values are then interpolated to create a continuous extension of the numerical solutions on $[0, s]$. The resulting interpolants are then used on the next interval, $[s, 2s]$, to evaluate the functions $F(u(x, t-s), v(x, t-s))$ and $G(u(x, t-s), v(x, t-s))$. When starting the computation on the the second delay interval, we start off using the correct initial condition so that the solution itself is continuous at $t = s$. The solver then stops exactly at $t = 2s$ and the entire process is repeated. This implementation of the method of steps allows us to approximate the solutions of (3.2) numerically. The code described here can be found in the Appendix in §A.3.

3.4.2 Transient Method of Lines (Reverse Method of Lines)

In this section we discuss another numerical technique we have considered to approximate the solution to (3.2) numerically. This was actually the first technique we considered and spent months trying to implement it before realizing we could use the method of steps in Comsol as explained in §3.4.1. The main advantage of this method is that it may allow for error control in time and space. When solving a

time dependent PDE in Comsol there is an adaptive time stepping algorithm which allows for error control in time. In contrast, there is no error control in space for time dependent PDEs. The spatial mesh is chosen based on the geometry and the mesh remains the same during the entire computation. There would need to be an adaptive mesh algorithm to also have error control in space. Comsol does have an adaptive spatial mesh algorithm but only for PDEs that are stationary (no time dependance).

We have taken some initial steps in implementing the reverse method of lines (discretization in time first) which results in a system of stationary PDEs. The resulting system of stationary PDEs can be solved in Comsol. Moreover, we have the option of spatial adaptivity in Comsol and therefore the option of error control in space. We would need to write our own time stepping algorithm with error control to get error control in time.

We begin by replacing the time derivatives in (3.2) with the implicit Euler finite difference scheme, $\frac{\partial u}{\partial t} = \frac{u_j - u_{j-1}}{h_j}$. Here $h_j = t_j - t_{j-1}$ is the jth time step size while u_j and v_j are the approximations to $u(x, t_j)$ and $v(x, t_j)$ respectively. After simplifying we obtain

$$h_j \Delta_x u_j = (\tau_u + h_j \alpha_1^2 \varepsilon) u_j - \tau_u u_{j-1}, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \tag{3.40a}$$

$$\partial_{nx} u_j = 0, \quad x \in \partial\Omega_1 \tag{3.40b}$$

$$\varepsilon \partial_{nx} u_j = F(u_{j_T}, v_{j_T}), \quad x \in \partial\Omega_{\varepsilon_1} \tag{3.40c}$$

$$u_j = u_C, \quad t \leq 0 \tag{3.40d}$$

$$h_j \Delta_x v_j = (\tau_v + h_j \alpha_2^2 \varepsilon) v_j - \tau_v v_{j-1}, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \tag{3.40e}$$

$$\partial_{nx} v_j = 0, \quad x \in \partial\Omega_1 \tag{3.40f}$$

$$\varepsilon \partial_{nx} v_j = G(u_{j_T}, v_{j_T}), \quad x \in \partial\Omega_{\varepsilon_2} \tag{3.40g}$$

$$v_j = v_C, \quad t \leq 0. \tag{3.40h}$$

Here we have used the notation $u_{j_T} \equiv u(x, t_j - T)$ and so on. We assume the history data for u and v are the constants u_C and v_C. For now we will assume that we take uniform constant time steps $h_j = h$. We also assume that the time step can never be larger than the delay T. We will also assume that the step size h evenly divides into

T. This means that the delayed arguments, $t_j - T$, will access past solution values which were previous step points in time.

For each time step, we need to solve the BVP in (3.40). For $t \in [0, T]$ the boundary conditions (3.40c) and (3.40g) will be the constants $F(u_C, v_C)$ and $G(u_c, v_c)$ respectively. After each time step we need to save the solution in order to evaluate the delayed arguments later in time. For example, when we get to $t = T + h$ the delayed arguments will be accessing the solution at $t = h$ which we will have saved. We only need to save solution data back T units in time. As described in §3.4.1, we only need to save the solution values for u and v at two points in the spatial domain (one point on each compartment for both u and v). Therefore, after each time step we save four constant solution values and store them in four vectors. These four vectors are always of length T/h because we are always removing an entry and adding an entry after each time step. Using this setup, the boundary conditions (3.40c) and (3.40g) will be constant for every time step.

For each time step we need to have the entire solution from the previous step stored. This is because the implicit Euler method is a one step method and u_j depends on u_{j-1}, as seen in (3.40a). After solving for u_{j-1} in a given step, we then evaluate u_{j-1} at numerous points in the domain and write the data to a text file. The resulting file is very large file because it consists of x, y, z and $u_{j-1}(x, y, z)$ values for thousands of points in the spatial domain. When solving for u_j we need to have u_{j-1} as a function for input into the right hand side of (3.40a). This is accomplished by reading the text file and then interpolating the data with a built in sub routine in Comsol. The result is a 3D function which gets passed to the right hand side of (3.40a). We do the same process for v_{j-1} to compute v_j. The process of saving the current solution at points over the entire domain and creating an accurate interpolant takes up a lot of time relative to the time needed to actually solve the stationary PDEs at every time step. There is not an efficient method in Comsol in which to store the solution from a stationary PDE computation and then use it for input into another PDE computation. This is in contrast to the time dependent problems where a current solution can be used as an initial condition to another computation quite efficiently (see §3.4.1). This is one of the major draw backs of implementing this method. For example, to advance through one time step it may take approximately

20 seconds. This is too long when using a constant step size. It only takes around 3-5 seconds to actually solve the PDE system. It is the storing and interpolation of the 3D space dimensional function that takes up all the computational time.

In the above description we have used a constant step size. This does not allow for any error control in time. To have error control in time we need some way of estimating the error for every time step. One way this can be done is by redoing the calculation with a higher order method such as BDF2. For each time step we can use BDF1 (implicit Euler) to get a first approximation and then BDF2 to get a second more accurate approximation. Then we can estimate the error at every step and reduce the step size accordingly depending on the required accuracy.

The variable time step BDF2 method coupled with the BDF1 method would allow for the possibility of adaptive error control in time. In order to do so we would need a away of estimating the error. Comsol has built in routines which can integrate expressions over the entire domain. These routines could be used to estimate the error between the approximations obtained from BDF1 and BDF2. For example, we could use some norm which involves an integral and the difference of the two approximations. Once the error is estimated, we would need to decide on an algorithm which changes the step size in some manner. This would give us error control in time. Then we could get error control in space by using the adaptive mesh refinement option in Comsol for solving the stationary PDEs at every time step.

There are several reasons why we have not pursued the reverse method of lines with error control in time and space. One reason, which was previously discussed, is that the method is currently very slow. Another reason is that we already have a much easier implementation of the method of steps in Comsol. Finally, the last reason is that we have had some numerical issues with using the reverse method of lines when the time step is small compared to ε. For example, if the time step is $h = O(\varepsilon^2)$ then there is a lot of error introduced into the approximation. We believe the spatial error is dominating the temporal error for time steps that are much smaller than ε. The error is even present when $h = O(\varepsilon)$. We give one possible explanation for this.

The diffusion coefficient in (3.40) is the time step h. Therefore a small time step results in a small diffusion coefficient in (3.40). The spatial mesh in Comsol

is constructed based on the geometry and is not adaptive by default for stationary problems. The size of ε will affect the mesh and the size of its elements. We think that a very small time step (diffusion coefficient) will result in sharper gradients and that the mesh element sizes, based on the size of ε, will not be small enough to solve the stationary problems accurately enough. We believe the spatial error is dominating the temporal error when the time step is $O(\varepsilon)$. We tried to resolve this issue by using the adaptive mesh refinement option which is available for solving stationary problems, but we still got the same results. We have also tried adjusting manually the mesh properties such as min/max element sizes, etc. We believe the problem can be resolved with enough tweaking but it is not something that we have pursued. The implicit Euler method code described here can be found in in the Appendix in §A.4.

3.4.3 Numerical Solution of DDAEs

In §3.3 we derived the following asymptotic time dependent approximations of (3.2) when $s = T/\varepsilon$,

$$\begin{aligned} u(x,\tau) &= u_0(\tau) + \varepsilon \left(\frac{c_1(\tau)}{|x - x_1|} + 4\pi c_1(\tau) R_n(x; x_1) + \chi_1(\tau) \right) \\ v(x,\tau) &= v_0(\tau) + \varepsilon \left(\frac{c_2(\tau)}{|x - x_2|} + 4\pi c_2(\tau) R_n(x; x_2) + \chi_2(\tau) \right). \end{aligned} \tag{3.41}$$

The time dependent functions here are found by solving the following system of DDAEs which are also found in §3.3. We write them here together for easy reference,

$$\begin{aligned} \begin{pmatrix} \frac{\tau_u}{3} u_0 \\ \frac{\tau_v}{3} v_0 \end{pmatrix}' &= - \begin{pmatrix} \frac{\alpha_1^2}{3} & 0 \\ 0 & \frac{\alpha_2^2}{3} \end{pmatrix} \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} + \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \\ \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} &= \begin{pmatrix} F(u_{0_T} + c_{1_T}, v_{0_T}) \\ G(u_{0_T}, v_{0_T} + c_{2_T}) \end{pmatrix} \\ \begin{pmatrix} \frac{\tau_u}{3} \chi_1 \\ \frac{\tau_v}{3} \chi_2 \end{pmatrix}' &= - \begin{pmatrix} \frac{\alpha_1^2}{3} & 0 \\ 0 & \frac{\alpha_2^2}{3} \end{pmatrix} \begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} + \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \\ \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} &= \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}_T + \begin{pmatrix} F_u & F_v \\ G_u & G_v \end{pmatrix}_T \begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix}_T + \begin{pmatrix} F_u & 0 \\ 0 & G_v \end{pmatrix}_T \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}_T . \end{aligned} \tag{3.42}$$

The functions B_1 and B_2 are defined in (3.38). In this section we discuss how to approximate the solutions of (3.42) numerically using Matlab.

In the simpler case where the partial derivatives F_u and G_v are zero then the above system can be written as a system of four DDEs. The implicit algebraic equations become explicit and can be isolated for c_1, c_2, A_1, and A_2. These equations are then substituted in to the differential equations. The result is a system of four DDEs for u_0, v_0, χ_1, and χ_2. These solutions could be approximated numerically by a solver such as **dde23** in Matlab.

In the more general case that $F = F(u, v)$ and $G = G(u, v)$, we can not rewrite the DDAEs in (3.42) as a system of DDEs for just the four variables u_0, v_0, χ_1, and χ_2. This is because we can not isolate for c_1, c_2, A_1, and A_2 as functions of just u_0, v_0, χ_1, and χ_2. In (3.42), when we substitute the algebraic equations into the differential equations the result is four DDEs and 8 unknowns. To use a DDE solver there needs to be an additional DDE for each of c_1, c_2, A_1, and A_2. We could obtain these additional DDEs by differentiating the algebraic equations in (3.42). The result though is a system of four neutral DDEs which are difficult to approximate numerically. We have tried using the neutral delay solver **ddensd** in Matlab to solve this system but it does not provide the results we need in all cases considered, since in general the functions c_1 and c_2 are discontinuous at $t = 0$. We will explain why this is shortly. **ddensd** approximates solutions of neutral DDEs by solving an approximating DDE that is not neutral. The approximating equation has a solution that is not only continuous, but the order of jumps decreases as the integration proceeds. For this reason the solution returned by **ddensd** is continuous. There is no way to get genuine jump discontinuities in the approximation produced by **ddensd** [88].

One method we have implemented to solve the DDAEs is the method of steps combined with a regular ODE solver in Matlab. First we substitute the four algebraic equations from (3.42) into the differential equations in (3.42). This leads to a system of 4 DDEs for u_0, v_0, χ_1, and χ_2. Although there are other functions in the DDEs, c_1, c_2, A_1, and A_2, they are all delayed. Therefore, for any given time step the solver only needs to access the history of these functions. If history is provided for these functions, as well as the functions u_0, v_0, χ_1, and χ_2, then every time delayed

function is known on the right hand side of the DDEs. We first provide history for all 8 unknowns. Then we solve a system of ODEs for u_0, v_0, χ_1, and χ_2 on the interval $[0, T]$. Once the solution on $[0, T]$ is obtained then we save this solution data. On the next interval, $[T, 2T]$, a system of ODEs is solved again and the time delayed function values are obtained by interpolating the stored solution on the previous interval. This is the method of steps as described in §3.4.1. We take the same precautions here as in §3.4.1 when stepping through the discontinuities. We include points of discontinuity as mesh points and make sure that the solver does not step over the discontinuities. After stepping to each discontinuity we restart the computation with a relatively small time step and low order method.

Because of the constant restarting, the method of steps implemented numerically is slower than a more advanced typical DDE solver such as **dde23**. Although slow, the method of steps code we have implemented does provide accurate numerical approximations to the system in (3.42). In §3.5 the numerical approximations of (3.42) which are used in (3.41), are compared with the full numerical approximations of the PDE system (3.2). The method of steps code described here can be found in the Appendix in §A.5.

We will end this section by considering the history for the time dependent functions in (3.42). Recall that $u(x, \tau) = u_h$ and $v(x, \tau) = v_h$ for $\tau \leq 0$. We will choose u_h and v_h to be constant in time and space. From the expressions in (3.41), it therefore seems reasonable to assume that $c_1 = c_2 = 0$ for $\tau < 0$. This is because these functions multiply spatial terms. As well, we set $\chi_1 = \chi_2 = 0$ for $\tau \leq 0$. The history for $u_0(\tau)$ and $v_0(\tau)$ is then u_h and v_h respectively. The functions A_1 and A_2 do not appear in (3.41) but they are multiplied by spatial terms in the higher order inner solutions in (3.35). Therefore we also require that $A_1 = A_2 = 0$ for $\tau \leq 0$.

Note that we have not defined the values for $c_1(0)$ and $c_2(0)$. These are determined by substituting $\tau = 0$ into the equations for c_1 and c_2 in (3.42). This gives $c_1(0) = F(u_h, v_h)$ and $c_2(0) = G(u_h, v_h)$. Similarly, $A_1(0)$ and $A_2(0)$ can be determined from (3.37) and it follows that $A_1(0) = A_2(0) = 0$. Depending on the choice of F and G, in general c_1 and c_2 will have a jump discontinuity at $\tau = 0$. It is usually the case that solutions to DDEs have derivative discontinuities but here we mean a discontinuity in the actual functions. The equations for c_1 and c_2 are actually

$$c_1(\tau) = \begin{cases} 0 & \text{if} \quad \tau < 0 \\ F(u_{0_T} + c_{1_T}, v_{0_T}) & \text{if} \quad \tau \geq 0 \end{cases}$$

$$c_2(\tau) = \begin{cases} 0 & \text{if} \quad \tau < 0 \\ G(u_{0_T}, v_{0_T} + c_{2_T}) & \text{if} \quad \tau \geq 0 \end{cases}$$

As a consequence, the correction terms (terms multiplied by ε) in the asymptotic expansions in (3.41) are discontinuous at $\tau = 0$. Furthermore, these discontinuities propagate because of the delay. In contrast, the solutions of the PDEs in (3.2) are continuous in time. This unusual discontinuity in the asymptotic solution is just a result of using constant history for the PDEs. We chose constant history since it was the easiest to implement numerically. The discontinuities smooth out as time evolves and eventually become non existent after several delay intervals have passed. This can be seen from the numerical simulations in Figure 3.3. As well, the discontinuities in c_1 and c_2 actually capture the behavior that takes place for early time in the PDEs.

Since we have assumed u_h and v_h are constant in time and space there will be locations in the domain where the solutions of (3.2) change rapidly for early time. Immediately after $t = 0$ there are initial transients that occur where the solutions of (3.2) change from having a constant spatial profile to that of a spatial profile similar to a Green's function. These transients occur quite rapidly and only occur near the compartments. This is because the solutions of (3.2) are relatively constant in space away from the compartments. For example, $u(x, t)$ will have a sudden jump right after $t = 0$ but only in its inner region, $|x - x_1| \approx \varepsilon$. A similar thing happens to v but for $|x - x_2| \approx \varepsilon$. The way in which the asymptotic time dependent approximations in (3.41) capture this behavior is through the functions $c_1(\tau)$ and $c_2(\tau)$ and their respective jump discontinuities at $\tau = 0$.

3.5 Examples

In §3.3, the system (3.2) was considered for the case of large delay, $s = T/\varepsilon$. The main results were the leading order eigenvalue problem in (3.18) and the system of DDAEs in (3.42). In this section we will use the eigenvalue problem to find bifurcations in the DDAEs. We will show through numerical simulations that the same bifurcations occur in the full PDE system (3.2). Simultaneously, we will compare the output

between the numerical simulations of the DDAEs and the full PDE system. This is done by comparing the time dependent asymptotic approximations in (3.41) with the numerical simulations of (3.2).

3.5.1 Bistability

In this example we will consider a simpler case when $F_u = G_v = 0$. The functions we choose are

$$F(v) = \frac{2}{1+v^3}$$
$$G(u) = \frac{5/2}{1+u^3}.$$

For the parameters we choose $k_1 = 3/4$, $k_2 = 1$, $D_u = D_v = 1/3$, $R = 1$, $x_1 = (0.5, 0, 0)$, $x_2 = (-0.5, 0, 0)$, $\varepsilon = 0.01$, and leave T as a parameter. We will use different history values for different numerical simulations.

The leading order DDAE system in (3.34) becomes the following DDE system,

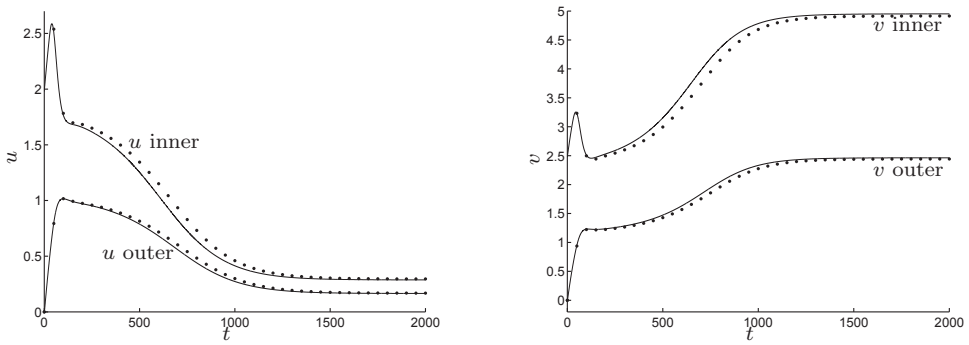
$$\begin{aligned} \frac{\partial u_0}{\partial \tau} &= -\frac{3}{4}u_0(\tau) + \frac{2}{1+v_0(\tau-T)^3} \\ \frac{\partial v_0}{\partial \tau} &= -v_0(\tau) + \frac{5/2}{1+u_0(\tau-T)^3}. \end{aligned} \tag{3.43}$$

The equilibria of this DDE system are the leading order equilibrium solutions of the full PDE system. They can be solved for numerically. The leading order equilibria are $(0.1623, 2.4893)$, $(1.0534, 1.1527)$, and $(2.6613, 0.12594)$. These are written in the form (u_0, v_0). To determine the stability of these constant equilibrium solutions we consider the nonlinear eigenvalue problem in (3.18). Recall that the partial derivatives in (3.18) are evaluated at the leading order equilibrium points. For a given equilibrium point and delay value T, we can solve for λ_1 numerically. There are multiples solutions for λ_1 since the eigenvalue problem is nonlinear. We are only interested in the eigenvalue which has the largest real part. The software package *Maple* is used to solve (3.18) for λ_1 with different values of T. In this discussion, when we refer to λ_1 we are referring to the eigenvalue with the largest real part.

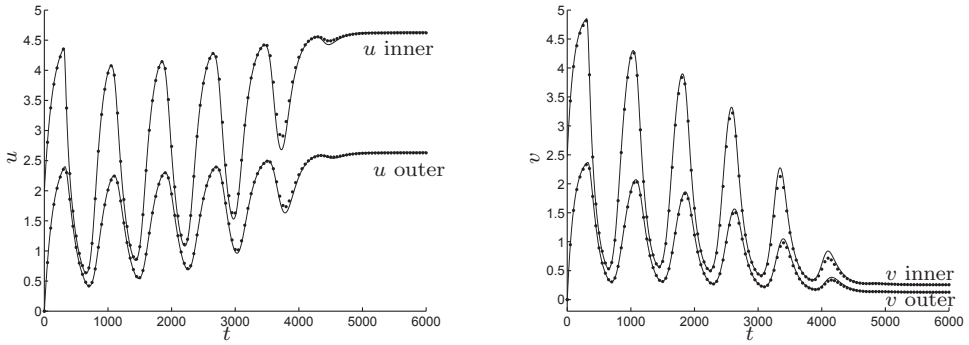
By computing the real part of λ_1 for multiple values of T, it can be concluded that the two equilibrium points, $(0.1623, 2.4893)$ and $(2.6613, 0.12594)$, are both stable for any choice of T. This is because the $\mathrm{Re}(\lambda_1)$ is negative for all T. The equilibrium point $(1.0534, 1.1527)$ is unstable for all T and this is because the $\mathrm{Re}(\lambda_1)$ is positive for

all T. For the equilibrium point $(1.0534, 1.1527)$, when $T < 1.839$ all the eigenvalues except for λ_1 have a negative real part. When T crosses through $T = 1.839$, the next eigenvalue with the second largest real part crosses through the imaginary axis. As T is increased, more of the eigenvalues cross the imaginary axis in sequence. Therefore as T increases more oscillations in the solutions of (3.43) can be observed. There is not an actual Hopf Bifurcation though and oscillations are not sustained. The solutions of (3.43) always tend to either one of the two constant stable equilibrium points as time evolves. Which equilibrium point the solution tends to depends on both the initial history as well as the value of T.

The behavior discussed above is apparent in the following numerical simulations. We emphasize that the dynamics of the DDEs in (3.43) reflect the dynamics of the more general PDE system (3.2). To solve the DDEs in (3.43) we will use the method of steps as outlined in §3.4.3. Not only will we be solving (3.43), but also the complete system of DDAEs in (3.42). To solve the PDEs in (3.2) we will also use the method of steps as outlined in §3.4.1.

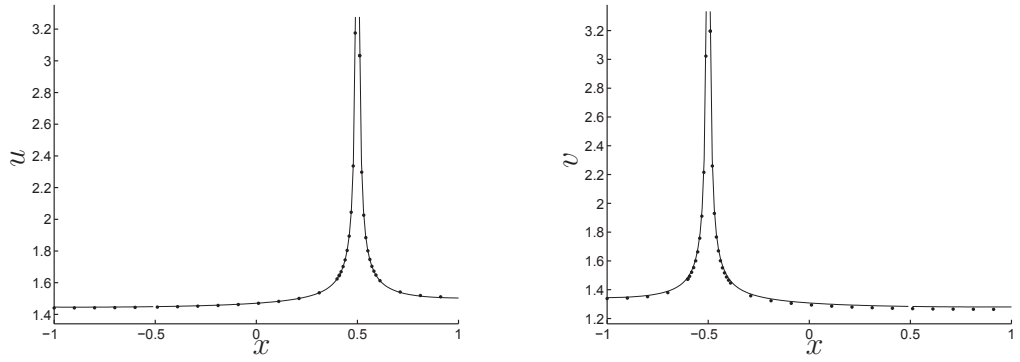


(a) $u_0 + \varepsilon u_1$ and $u(x,t)$ (b) $v_0 + \varepsilon v_1$ and $v(x,t)$



(c) $u_0 + \varepsilon u_1$ and $u(x,t)$ (d) $v_0 + \varepsilon v_1$ and $v(x,t)$

Figure 3.1: These figures use the history $u_H = 0$ and $v_H = 0$. In each figure the solid curve is an asymptotic solution from (3.41) obtained from solving (3.42). The dotted curve is the numerical solution of (3.2) obtained from using the method of steps described in §3.4.1. In Comsol a relative tolerance of 1e-9 and an absolute tolerance of 1e-9 was used. In Matlab a relative tolerance of 1e-7 and an absolute tolerance of 1e-7 was used. In each figure we plot u at the point (.51,0,0) which is in its inner region on the surface of its compartment. We also plot u in its outer region away from its compartment in each figure. We also repeat this for v except now we use the point (-.51,0,0) in its inner region. In figures 3.1(a) and 3.1(b) we use a delay value of $T = 0.25$. In figures 3.1(c) and 3.1(d) we use a delay value of $T = 3$.



(a) $u_0 + \varepsilon u_1$ and $u(x,t)$ (b) $v_0 + \varepsilon v_1$ and $v(x,t)$

Figure 3.2: These two figures show u and v along the x-axis where we have set $y = z = 0$. The solid curves are the asymptotic approximations from (3.41) and the dotted lines are the numerical solutions of (3.2) with the same tolerances used in Figure 3.1. The solutions here are plotted at $t = 900$ and the delay is $T = 3$. Note that the solutions are basically constant in space away from the two compartments which are located at $(-.5, 0, 0)$ and $(.5, 0, 0)$. This simulation is the same simulation used in figures 3.1(c) and 3.1(d)

We can see from Figure 3.1 that increasing the delay leads to more oscillations in the solutions. It is also interesting that although T does not affect the stability of the equilibrium points, it can change which of the stable equilibria the solution approaches. In Figure 3.1 the same initial history was used for the two simulations but a different delay value resulted in a different equilibrium solution being approached. This is because the increase in oscillations made the solution jump to the other stable equilibrium.

3.5.2 Hopf Bifurcation

In this example we will choose functions for F and G which give rise to a Hopf bifurcation. The functions $F(u,v)$ and $G(u,v)$ are chosen to be

$$F(u,v) = \frac{4u}{(1/2+u)(1+v^2)}$$
$$G(u,v) = \frac{2v}{1/4+v}u.$$

For the parameters we choose $k_1 = 1 = k_2 = 1$, $D_u = D_v = 1/3$, $R = 1$, $x_1 = (0.5, 0, 0)$, $x_2 = (-0.5, 0, 0)$, $\varepsilon = 0.01$, and leave T as the bifurcation parameter.

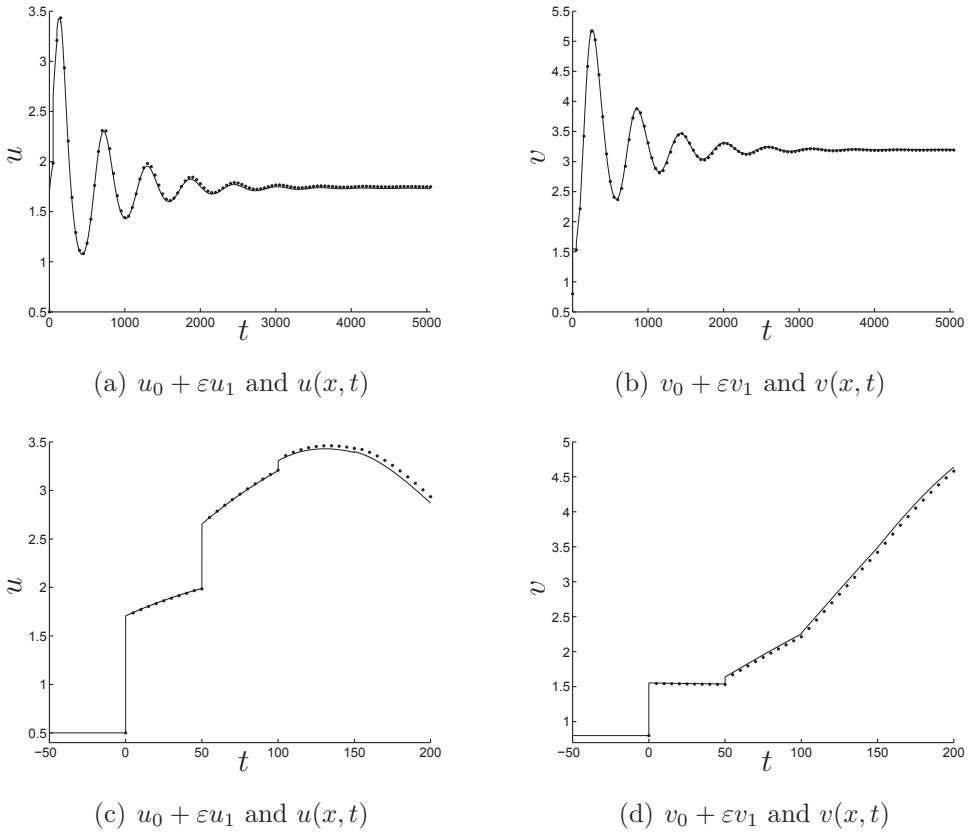
Again we are considering the case of large delay $s = T/\varepsilon$. For the history data we will choose $u_h = .5$ and $v_h = .8$.

These type of enzyme kinetic functions were studied in [47]. With this choice of F and G the model in (3.2) has both positive and negative feedback between the two signalling compartments. In [47] similar enzyme functions were analyzed and a Hopf-bifurcation was found which resulted in sustained oscillations. The system was much larger than two signalling proteins though. Because of the delay present in the model (3.2), sustained oscillations can be obtained in a much simpler system.

First we solve for the leading order equilibrium solutions of (3.2). These constants u_0 and v_0 are found from solving (3.3). The leading order equilibrium solution is $(u_0, v_0) = (0.8662, 1.6074)$. For this equilibrium point and different values of T, we can solve (3.18) numerically in *Maple* for λ_1. Here we will again assume that λ_1 is the eigenvalue with largest real part. We can also solve for critical T values which give rise to purely imaginary eigenvalues. For $T < 1.02332$ the real part of λ_1 is negative so the equilibrium solution is stable. This eigenvalue crosses the imaginary axis when $T = 1.02332$. For $T > 1.02332$ the real part of λ_1 is positive and therefore the equilibrium solution is unstable.

When $T < 1.02332$ the solutions have decaying oscillations which approach the stable equilibrium solution. For $T > 1.02332$, the equilibrium solution is unstable and a stable periodic orbit is formed. This Hopf bifurcation can be observed numerically in both (3.2) and (3.42). We again use the method of steps as described in §3.4.1 and §3.4.3.

For the first simulations we will use a delay value smaller than 1.02332.



(a) $u_0 + \varepsilon u_1$ and $u(x,t)$ (b) $v_0 + \varepsilon v_1$ and $v(x,t)$

(c) $u_0 + \varepsilon u_1$ and $u(x,t)$ (d) $v_0 + \varepsilon v_1$ and $v(x,t)$

Figure 3.3: All of these figures come from the same simulation in which $u_h = .5$, $v_h = .8$, and $T = 0.5$. In Figure 3.3(a) we plot u at the point (.51,0,0) which is in its inner region. The solid curve is the asymptotic solution (using the correction term). The dotted curve is the Comsol solution with relative tolerance of 1e-8, and an absolute tolerance of 1e-10. Figure 3.3(b) is v in its inner region at the point (-.51,0,0). Figure 3.3(c) is the plot from Figure 3.3(a) but on a different time interval. Figure 3.3(d) is the plot from Figure 3.3(b) but on a different time interval.

In §3.4.1 we discussed that constant initial history for (3.2) leads to c_1 and c_2 having jump discontinuities at $t = 0$. This is how the asymptotic DDAEs in (3.42) are able to capture the rapid jumps for the solutions in the inner regions. For the PDEs in (3.2), the solutions for u and v rapidly change from constant history at $t = 0$ to spatial profiles similar to Green's function. This sudden jump, at one point in space, is seen in figures 3.3(c) and 3.3(d). The jumps are propagated but eventually smooth out.

Now we consider numerical simulations for $T > 1.02332$. Therefore we expect to see sustained oscillations in the solutions of the PDEs and DDAEs. We reduce the the tolerances in Comsol to a relative tolerance of 1e-5, and an absolute tolerance of

1e-3 so that we can solve over a large number of delay periods.

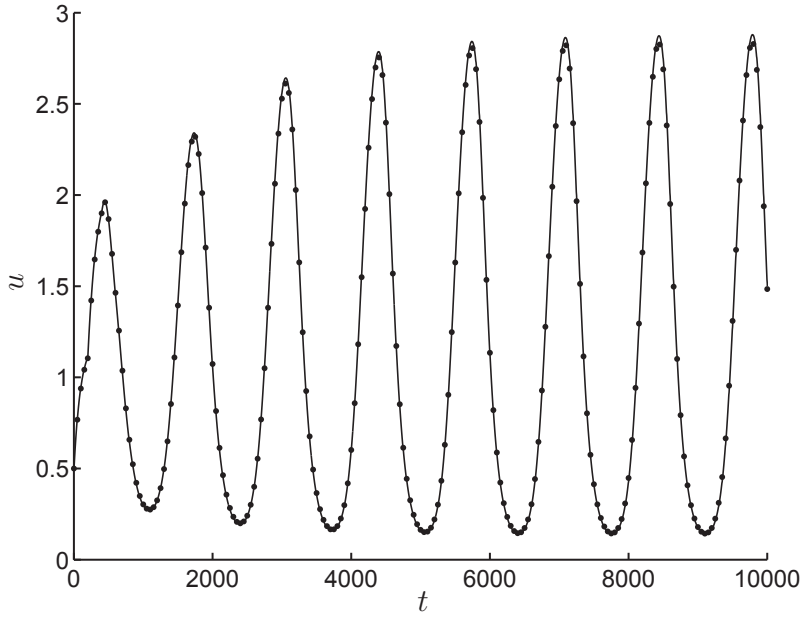


Figure 3.4: The solid curve is the asymptotic approximation (with correction) at the point (-1,0,0). The dotted curve is the numerical solution of $u(x,t)$ at the same point obtained from the method of steps in Comsol. The delay is $T = 2$ so the equilibrium solution is unstable and there is a stable periodic orbit.

On the time scale above, the agreement between the asymptotics and the PDE system is very good. In Figure 3.5 though, we show how the terms χ_1 and χ_2 blow up as $t \to \infty$. Therefore this leads to the asymptotic approximations from (3.39) blowing up. This growth is commonly referred to as secular growth. In general it occurs when a straight forward application of perturbation theory is applied to weakly nonlinearly problems with bounded oscillatory solutions. To get a better asymptotic approximation without the secular growth a more advanced approach is required. We will use the Poincaré-Lindstedt method in §3.6 to improve our asymptotic approximation.

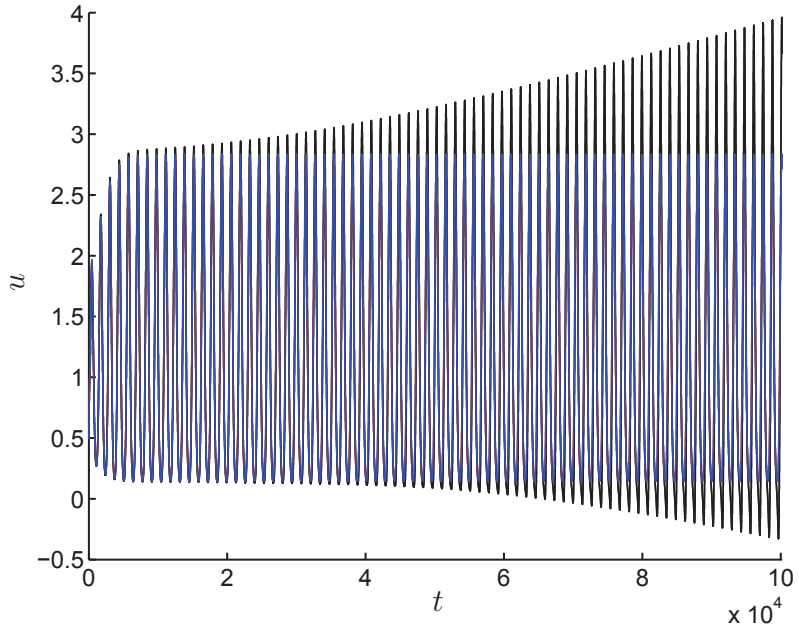


Figure 3.5: This is the same plot as in Figure 3.4 but on a longer time scale. It shows how the asymptotic expansion (black) with the correction term is no longer valid as $t \to \infty$. The numerical solution of (3.2) (blue) does not increase without bound as $t \to \infty$ because the solution is periodic.

In the case of a Hopf bifurcation leading to sustained oscillations, the functions χ_1 and χ_2 increase without bound as $\tau \to \infty$. We have observed this in every Hopf bifurcation example we have considered. The solutions of (3.2) have sustained oscillations that do not grow in time. Therefore the DDAEs in (3.42) are not valid for later times in the case of a Hopf bifurcation leading to stable periodic orbits.

3.6 Poincaré-Lindstedt Method for the PDE Model with Delay

In §3.5.2 we showed an example with a Hopf bifurcation. For delay values greater than a critical value, sustained oscillations were observed in the numerical solutions of the PDE model as well as the approximating DDAEs. As time evolves, the oscillations in the PDE model do not grow and the amplitude remains constant in time. In the system (3.42), the functions χ_1 and χ_2 grow without bound as $\tau \to \infty$ and therefore the asymptotic approximations of (3.41) are not uniformly valid in time. In this section we use the Poincaré-Lindstedt method to derive a more accurate time dependent approximation to (3.2) in the case of a large delay leading to a Hopf

bifurcation.

This is a nice application of the Poincaré-Lindstedt method to a non standard problem (the PDE model in (3.2)). Although the analysis is relatively straight forward, the numerical implementation is a lot more interesting. A linear differential algebraic operator has to be constructed in the just the right way for everything to work. One of the main goals is finding periodic solutions in the kernel of the adjoint of this operator. We discretize the operator to turn it into a matrix and then use an eigenvalue solver to find the kernel of this discrete operator. The kernel of the discretized operator approximates the kernel of the continuous operator.

We begin by seeking periodic solutions of the system (3.2) near a Hopf bifurcation when $s = T/\varepsilon$. The idea is to rescale time as $\tau = \omega t$ where $\omega = \omega_0 + \omega_1\varepsilon + \omega_2\varepsilon^2 + \ldots$. Since we already had success in §3.3 with the time scale $\tau = \varepsilon t$, we will choose $\omega_0 = 0$. Without loss of generality we can choose $\omega_1 = 1$ since its value only affects the period of the oscillations in the rescaled DDAEs.

First we define a new time variable

$$\begin{aligned} \tau &= \omega t \\ \omega &= \varepsilon + \omega_2\varepsilon^2, \end{aligned} \tag{3.44}$$

and substitute this change of variables into (3.2). After rescaling, the PDE model with delay in (3.2) becomes

$$\begin{aligned} \tau_u\omega\frac{\partial u}{\partial \tau} &= \Delta_x u - \alpha_1^2\varepsilon u, \quad x \in \Omega_1\backslash\Omega_\varepsilon \\ \partial_{nx}u &= 0, \quad x \in \partial\Omega_1 \\ \varepsilon\partial_{nx}u &= F(u(x,\tau-\omega\frac{T}{\varepsilon}), v(x,\tau-\omega\frac{T}{\varepsilon})), \quad x \in \partial\Omega_{\varepsilon_1} \\ \\ \tau_v\omega\frac{\partial v}{\partial \tau} &= \Delta_x v - \alpha_2^2\varepsilon v, \quad x \in \Omega_1\backslash\Omega_\varepsilon \\ \partial_{nx}v &= 0, \quad x \in \partial\Omega_1 \\ \varepsilon\partial_{nx}v &= G(u(x,\tau-\omega\frac{T}{\varepsilon}), v(x,\tau-\omega\frac{T}{\varepsilon})), \quad x \in \partial\Omega_{\varepsilon_2}, \end{aligned} \tag{3.45}$$

If ω_2=0 then the following analysis is identical to §3.3. The analysis here is similar to that of §3.3 so we omit most of the details.

First we expand the outer and inner solutions, as in (3.29), and obtain the following rescaled outer problems,

$$\begin{aligned}
\Delta_x u_0 = 0, \quad x \in \Omega_1 \backslash \{x_1\}, & \qquad \partial_{nx} u_0 = 0, \quad x \in \partial\Omega_1 \\
\tau_u \frac{\partial u_0}{\partial \tau} = \Delta_x u_1 - \alpha_1^2 u_0, \quad x \in \Omega_1 \backslash \{x_1\}, & \quad \partial_{nx} u_1 = 0, \quad x \in \partial\Omega_1 \\
\tau_u \frac{\partial u_1}{\partial \tau} + \tau_u \omega_2 \frac{\partial u_0}{\partial \tau} = \Delta_x u_2 - \alpha_1^2 u_1, \quad x \in \Omega_1 \backslash \{x_1\}, & \quad \partial_{nx} u_2 = 0, \quad x \in \partial\Omega_1
\end{aligned}$$

$$\begin{aligned}
\Delta_x v_0 = 0, \quad x \in \Omega_1 \backslash \{x_2\}, & \qquad \partial_{nx} v_0 = 0, \quad x \in \partial\Omega_1 \\
\tau_v \frac{\partial v_0}{\partial \tau} = \Delta_x v_1 - \alpha_2^2 v_0, \quad x \in \Omega_1 \backslash \{x_2\}, & \quad \partial_{nx} v_1 = 0, \quad x \in \partial\Omega_1 \\
\tau_v \frac{\partial v_1}{\partial \tau} + \tau_v \omega_2 \frac{\partial v_0}{\partial \tau} = \Delta_x v_2 - \alpha_2^2 v_1, \quad x \in \Omega_1 \backslash \{x_2\}, & \quad \partial_{nx} v_2 = 0, \quad x \in \partial\Omega_1.
\end{aligned}$$

The inner problems are

$$\begin{aligned}
0 &= \Delta_y u_0^{(i)}, \quad \rho > 1 \\
-\partial_\rho u_0^{(i)} &= F(u_{0_T}^{(i)}, v_{0_T}), \quad \rho = 1 \\
0 &= \Delta_y u_1^{(i)}, \quad \rho > 1 \\
-\partial_\rho u_1^{(i)} &= F_u(u_{0_T}^{(i)}, v_{0_T}) u_{1_T}^{(i)} + F_v(u_{0_T}^{(i)}, v_{0_T}) v_{1_T} \\
&\quad - T\omega_2 \left(F_u(u_{0_T}^{(i)}, v_{0_T}) u_{0_T}^{(i)\prime} + F_v(u_{0_T}^{(i)}, v_{0_T}) v_{0_T}{}' \right), \quad \rho = 1 \\
0 &= \Delta_y v_0^{(i)}, \quad \rho > 1 \\
-\partial_\rho v_0^{(i)} &= G(u_{0_T}, v_{0_T}^{(i)}), \quad \rho = 1 \\
0 &= \Delta_y v_1^{(i)}, \quad \rho > 1 \\
-\partial_\rho v_1^{(i)} &= G_u(u_{0_T}, v_{0_T}^{(i)}) u_{1_T} + G_v(u_{0_T}, v_{0_T}^{(i)}) v_{1_T}^{(i)} \\
&\quad - T\omega_2 \left(G_u(u_{0_T}, v_{0_T}^{(i)}) u_{0_T}{}' + G_v(u_{0_T}, v_{0_T}^{(i)}) v_{0_T}^{(i)\prime} \right), \quad \rho = 1,
\end{aligned}$$

where $'$ denotes differentiation with respect to τ. The terms with a subscript T are delayed throughout this section meaning that $u_{0_T} \equiv u_0(\tau - T)$.

Similar to §3.3, we will derive two systems of DDAEs. A leading order system and and a correction system. The leading order system here is the same as (3.34). Just as in §3.3, we can write out the asymptotic time dependent solutions of (3.45) as

$$\begin{aligned}
u(x,\tau) &= u_0(\tau) + \varepsilon \left(4\pi c_1(\tau) G_n(x; x_1) + \chi_1(\tau) \right) + O(\varepsilon^2) \\
v(x,\tau) &= v_0(\tau) + \varepsilon \left(4\pi c_2(\tau) G_n(x; x_2) + \chi_2(\tau) \right) + O(\varepsilon^2).
\end{aligned}$$

Recall that the leading order DDAEs for u_0, v_0, c_1 and c_2 are

$$\begin{aligned}\begin{pmatrix}\frac{\tau_u}{3}u_0\\ \frac{\tau_v}{3}v_0\end{pmatrix}' &= -\begin{pmatrix}\frac{\alpha_1^2}{3} & 0\\ 0 & \frac{\alpha_2^2}{3}\end{pmatrix}\begin{pmatrix}u_0\\ v_0\end{pmatrix}+\begin{pmatrix}c_1\\ c_2\end{pmatrix}\\ \begin{pmatrix}c_1\\ c_2\end{pmatrix} &= \begin{pmatrix}F(u_{0_T}+c_{1_T}, v_{0_T})\\ G(u_{0_T}, v_{0_T}+c_{2_T})\end{pmatrix}.\end{aligned} \tag{3.46}$$

Here we are assuming there exists a critical value of T which leads to a Hopf bifurcation. Furthermore, we assume there are stable periodic solutions to the leading order system (3.46) for values of T greater than the critical delay value. Therefore $\begin{pmatrix}u_0\\ v_0\end{pmatrix}$ and $\begin{pmatrix}c_1\\ c_2\end{pmatrix}$ are periodic. We will denote the period as P.

The parameter ω_2 appears in the system of DDAEs for χ_1, χ_2, A_1, and A_2. This correction system is

$$\begin{aligned}\begin{pmatrix}\frac{\tau_u}{3}\chi_1\\ \frac{\tau_v}{3}\chi_2\end{pmatrix}' &= -\begin{pmatrix}\frac{\alpha_1^2}{3} & 0\\ 0 & \frac{\alpha_2^2}{3}\end{pmatrix}\begin{pmatrix}\chi_1\\ \chi_2\end{pmatrix}+\begin{pmatrix}A_1\\ A_2\end{pmatrix}-\omega_2\begin{pmatrix}\frac{\tau_u}{3}u_0\\ \frac{\tau_v}{3}v_0\end{pmatrix}'\\ \begin{pmatrix}A_1\\ A_2\end{pmatrix} &= \begin{pmatrix}B_1\\ B_2\end{pmatrix}_T+\begin{pmatrix}F_u & F_v\\ G_u & G_v\end{pmatrix}_T\begin{pmatrix}\chi_1\\ \chi_2\end{pmatrix}_T+\begin{pmatrix}F_u & 0\\ 0 & G_v\end{pmatrix}_T\begin{pmatrix}A_1\\ A_2\end{pmatrix}_T-T\omega_2\begin{pmatrix}c_1\\ c_2\end{pmatrix}'.\end{aligned} \tag{3.47}$$

Note that we have simplified the two terms in the equations for A_1 and A_2 which are multiplied by $T\omega_2$,

$$\begin{aligned}c_1'(\tau) &= F_{u_T}(u_{0_T}'+c_{1_T}')+F_{v_T}v_{0_T}'\\ c_2'(\tau) &= G_{u_T}u_{0_T}'+G_{v_T}(v_{0_T}'+c_{2_T}').\end{aligned}$$

The functions B_1 and B_2 are defined as

$$\boldsymbol{B} = \begin{pmatrix}B_1(\tau)\\ B_2(\tau)\end{pmatrix} = \begin{pmatrix}F_u 4\pi c_1 R_n(x_1;x_1)+F_v 4\pi c_2 G_n(x_1;x_2)\\ G_u 4\pi c_1 G_n(x_2;x_1)+G_v 4\pi c_2 R_n(x_2;x_2)\end{pmatrix}.$$

The DDAEs in (3.47) differ from the corresponding system in (3.37) because of the now present ω_2 term. The parameter ω_2 is chosen to remove the secular growth. The functions u_0', v_0', c_1', and c_2' are each multiplied by ω_2 in (3.47) and are referred to as secular terms. When ω_2 was zero, as in (3.37), it was not possible to find a periodic solution for χ_1 and χ_2. The goal now is to formulate a solvability condition for ω_2 which will lead to the solutions of (3.47) being periodic.

With some rearranging we can rewrite (3.47) as

$$\begin{pmatrix} \frac{\tau_u}{3} & 0 & 0 & 0 \\ 0 & \frac{\tau_v}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \chi_1 \\ \chi_2 \\ A_1 \\ A_2 \end{pmatrix}' - \begin{pmatrix} F_u & F_v & F_u & 0 \\ G_u & G_v & 0 & G_v \\ F_u & F_v & F_u & 0 \\ G_u & G_v & 0 & G_v \end{pmatrix}_T \begin{pmatrix} \chi_1 \\ \chi_2 \\ A_1 \\ A_2 \end{pmatrix}_T + \begin{pmatrix} \frac{\alpha_1^2}{3} & 0 & 0 & 0 \\ 0 & \frac{\alpha_2^2}{3} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \chi_1 \\ \chi_2 \\ A_1 \\ A_2 \end{pmatrix} = \begin{pmatrix} -\omega_2 \left(\frac{\tau_u}{3} u_0' + T c_1' \right) + B_{1_T} \\ -\omega_2 \left(\frac{\tau_v}{3} v_0' + T c_2' \right) + B_{2_T} \\ B_{1_T} - T\omega_2 c_1' \\ B_{2_T} - T\omega_2 c_2' \end{pmatrix}.$$

Note that we have substituted the equations for A_1 and A_2 from (3.47) directly into the differential equations for χ_1 and χ_2. The reason for this is discussed later in Chapter 5.

From observing the left hand side of the above equation, we define the linear operator $\mathcal{L}$,

$$\mathcal{L}\boldsymbol{Y} \equiv \boldsymbol{\Omega}\boldsymbol{Y}' - \boldsymbol{J_2}_T \boldsymbol{Y}_T + \boldsymbol{K}\boldsymbol{Y}. \tag{3.48}$$

Here, $\boldsymbol{Y}$ is a vector valued function of length four. The matrices in (3.48) are defined as

$$\boldsymbol{\Omega} = \begin{pmatrix} \frac{\tau_u}{3} & 0 & 0 & 0 \\ 0 & \frac{\tau_v}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \boldsymbol{J_2} = \begin{pmatrix} F_u & F_v & F_u & 0 \\ G_u & G_v & 0 & G_v \\ F_u & F_v & F_u & 0 \\ G_u & G_v & 0 & G_v \end{pmatrix}, \boldsymbol{K} = \begin{pmatrix} \frac{\alpha_1^2}{3} & 0 & 0 & 0 \\ 0 & \frac{\alpha_2^2}{3} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.49}$$

If we define the vector valued function $\boldsymbol{X} = (\chi_1, \chi_2, A_1, A_2)^\top$ then we can write the system for χ_1, χ_2, A_1, and A_2 as

$$\mathcal{L}\boldsymbol{X} = \begin{pmatrix} -\omega_2 \left(\frac{\tau_u}{3} u_0' + T c_1' \right) + B_{1_T} \\ -\omega_2 \left(\frac{\tau_v}{3} v_0' + T c_2' \right) + B_{2_T} \\ B_{1_T} - T\omega_2 c_1' \\ B_{2_T} - T\omega_2 c_2' \end{pmatrix}. \tag{3.50}$$

We first note that the derivative of the leading order solution vector,

$$\left(u_0', v_0', c_1', c_2' \right)^\top,$$

is in the kernel of the operator $\mathcal{L}$. These terms show up on the right hand side of (3.50) and are the reason for the secular growth. To find a periodic solution of (3.50), without secular growth, we require the right hand side of (3.50) to be orthogonal to the kernel of the adjoint of $\mathcal{L}$. This is an application of the Fredholm Alternative. The next step is to find the adjoint operator $\mathcal{L}^*$.

First we define the standard inner product used on the space of real vector valued functions with period P,

$$\langle \boldsymbol{Y}, \boldsymbol{Z} \rangle = \int_0^P \boldsymbol{Y}^\top \boldsymbol{Z}\, d\tau. \tag{3.51}$$

Here we use $\top$ to denote the transpose operator and assume that $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are both periodic with period P. Then we have

$$\begin{aligned}
\langle \mathcal{L}\boldsymbol{Y}, \boldsymbol{Z} \rangle &= \int_0^P (\boldsymbol{\Omega}\boldsymbol{Y}' - \boldsymbol{J_2}_T \boldsymbol{Y}_T + \boldsymbol{K}\boldsymbol{Y})^\top \boldsymbol{Z}\, d\tau, \quad \text{Definition of } \mathcal{L} \text{ and } \langle , \rangle . \\
&= \int_0^P \boldsymbol{Y}'^\top \boldsymbol{\Omega}\boldsymbol{Z} - \boldsymbol{Y}_T^\top \boldsymbol{J_2}_T^\top \boldsymbol{Z} + \boldsymbol{Y}^\top \boldsymbol{K}\boldsymbol{Z}\, d\tau, \quad \text{Properties of transpose.} \\
&= \int_0^P -\boldsymbol{Y}^\top \boldsymbol{\Omega}\boldsymbol{Z}' - \boldsymbol{Y}_T^\top \boldsymbol{J_2}_T^\top \boldsymbol{Z} + \boldsymbol{Y}^\top \boldsymbol{K}\boldsymbol{Z}\, d\tau, \text{ Integration by parts and periodicity.} \\
&= \int_0^P -\boldsymbol{Y}^\top \boldsymbol{\Omega}\boldsymbol{Z}' - \boldsymbol{Y}^\top \boldsymbol{J_2}^\top \boldsymbol{Z}_+ + \boldsymbol{Y}^\top \boldsymbol{K}\boldsymbol{Z}\, d\tau, \quad \text{Periodicity.} \\
&= \int_0^P \boldsymbol{Y}^\top \left(-\boldsymbol{\Omega}\boldsymbol{Z}' - \boldsymbol{J_2}^\top \boldsymbol{Z}_+ + \boldsymbol{K}\boldsymbol{Z}\right) d\tau, \\
&= \langle \boldsymbol{Y}, \mathcal{L}^*\boldsymbol{Z} \rangle .
\end{aligned}$$

Thus we have found the adjoint operator $\mathcal{L}^*$. It is defined as

$$\mathcal{L}^*\boldsymbol{Z} \equiv -\boldsymbol{\Omega}\boldsymbol{Z}' - \boldsymbol{J_2}^\top \boldsymbol{Z}_+ + \boldsymbol{K}\boldsymbol{Z}. \tag{3.52}$$

Here we use $+$ as a subscript to mean $\boldsymbol{Z}_+ \equiv \boldsymbol{Z}(\tau + T)$. It does the opposite of the delay, $\boldsymbol{Z}_T \equiv \boldsymbol{Z}(\tau - T)$. In general we will refer to the $+$ in $\boldsymbol{Z}_+$ as the advancement operator.

Next we find the solvability condition for ω_2. We first let $\boldsymbol{Y}$ be an arbitrary vector valued function in the kernel of $\mathcal{L}^*$. Then the inner product of each side of (3.50) is

$$\left\langle \begin{pmatrix} -\omega_2 \left(\frac{\tau_u}{3} u_0' + T c_1'\right) + B_{1_T} \\ -\omega_2 \left(\frac{\tau_v}{3} v_0' + T c_2'\right) + B_{2_T} \\ B_{1_T} - T\omega_2 c_1' \\ B_{2_T} - T\omega_2 c_2' \end{pmatrix}, \boldsymbol{Y} \right\rangle = 0.$$

Letting $\boldsymbol{Y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$, we can solve the above equation for ω_2 to get

$$\omega_2 = \frac{\int_0^P B_{1_T}(y_1 + y_3) + B_{2_T}(y_2 + y_4)\, d\tau}{\int_0^P \frac{\tau_u}{3} u_0' y_1 + \frac{\tau_v}{3} v_0' y_2 + T c_1'(y_1 + y_3) + T c_2'(y_2 + y_4)\, d\tau}. \tag{3.53}$$

The equation in (3.53) can be solved numerically for ω_2. After solving for ω_2, this value is used in the system (3.50). With this value for ω_2 the solutions of (3.42) will be periodic. The most challenging part of using the formula in (3.53) is finding a non trivial periodic solution $\boldsymbol{Y}$ such that $\mathcal{L}^*\boldsymbol{Y} = 0$. This is the topic of the next section.

3.6.1 Computing the Kernel of $\mathcal{L}^*$

In this section we will describe the method we use to solve $\mathcal{L}^*(\boldsymbol{Y}) = 0$ where $\mathcal{L}^*$ is defined in (3.52). We seek a non trivial periodic solution, $\boldsymbol{Y}$, which is in the kernel of the adjoint operator. Any vector valued function in the kernel of the adjoint satisfies

$$\begin{aligned} -\boldsymbol{\Omega Y'} - \boldsymbol{J_2}^\top \boldsymbol{Y}_+ + \boldsymbol{K Y} &= 0 \\ \boldsymbol{Y}(0) &= \boldsymbol{Y}(P). \end{aligned} \tag{3.54}$$

The system in (3.54) consists of two differential equations, two algebraic equations with delay, and periodic boundary conditions. There is not a well known solver that we know of that can handle DDAEs with boundary conditions in general. Since we are looking for periodic solutions of (3.54) it is more convenient to use a boundary value problem solver as opposed to an initial value problem solver. Since $\mathcal{L}^*$ is linear we can use discretization in time to turn the the differential operator $\mathcal{L}^*$ into a matrix operator. We can then approximate the kernel of the continuous operator $\mathcal{L}^*$ with the kernel of the matrix.

The operator $\mathcal{L}^*$ acts on a vector valued function $\boldsymbol{Y}(\tau)$ of the form $\boldsymbol{Y}(\tau) = (y_1(\tau), y_2(\tau), y_3(\tau), y_4(\tau))^\top$. We begin by discretizing the time interval $[0, P)$ into $n+1$ points $\tau_0 < \tau_1 < \tau_2 < \ldots < \tau_n$ where $\tau_i = nh$ for $i = 0, 1, \ldots n$. We choose the step size h so that $(n+1)h = P$. We do not include $\tau = P$ as a mesh point in the

discretized interval because $Y(P) = Y(0)$ is already represented when $n = 0$. We discretize $\boldsymbol{Y}$ on this mesh by defining the vector

$$\begin{aligned}\overline{\boldsymbol{Y}} = (&y_1(0), y_1(h), \ldots, y_1(nh), y_2(0), y_2(h), \ldots, y_2(nh),\\ &y_3(0), y_3(h), \ldots, y_3(nh), y_4(0), y_4(h), \ldots, y_4(nh))^\top.\end{aligned} \tag{3.55}$$

The periodic boundary conditions imply that $y_j(0) = y_j((n+1)h)$ for $j = 1, 2, 3, 4$.

For notational convenience we define $y_{ji} \equiv y_j(ih)$ where $j = 1, 2, 3, 4$ and i=$0, 1, 2, \ldots, n$. The discretized vector, $\overline{\boldsymbol{Y}}$, is a vector of length $4(n+1)$. We also need to discretize the derivative operator, the advancement operator, and the other matrices in (3.54). The discretization process turns the continuous operator $\mathcal{L}^*$ into a $4(n+1)$ by $4(n+1)$ matrix which we will refer to as $\overline{\boldsymbol{L}}$. As n increases in size, the kernel of $\overline{\boldsymbol{L}}$ will converge to the kernel of $\mathcal{L}^*$.

To solve for the kernel of $\overline{\boldsymbol{L}}$ we will use an eigenvalue solver to find the zero eigenvalue whose corresponding eigenvector will be in the kernel. We use the *eigs* command in Matlab which allows to compute the smallest eigenvalues and corresponding eigenvectors. Since the matrix $\overline{\boldsymbol{L}}$ is of size $4(n+1)$ by $4(n+1)$ it is possible to run into many numerical issues in finding the eigenvalues for large n. In order to keep the size of the matrix as small as possible, without sacrificing accuracy, we use higher order methods to discretize the derivative and advancement operators. It may seem unnecessary but we end up getting faster convergence of the eigenvectors and therefore better convergence in computing ω_2. We also do not have to be concerned with finding the eigenvalues of relatively large matrices.

To discretize the derivative operator, $\frac{d}{d\tau}$, we use the following $O(h^4)$ derivative approximation for $y'(\tau)$,

$$y'(\tau) \approx \frac{y(\tau - 2h) - 8y(\tau - h) + 8y(\tau + h) - y(\tau + 2h)}{12h}.$$

We can use this derivative approximation to approximate $\boldsymbol{Y'}$ in (3.54). To begin with we will only consider the function $y_1(\tau)$ which is the first component of $\boldsymbol{Y}$. For example, $y_1'(0) \approx \frac{y_{1(-2)} - 8y_{1(-1)} + 8y_{11} - y_{12}}{12h} = \frac{y_{1(n-1)} - 8y_{1n} + 8y_{11} - y_{12}}{12h}$. Note that we have used the fact that $y_1(\tau)$ is periodic. For the function $y_1(\tau)$, which we discretized as the

vector $(y_{10}, y_{11}, \ldots, y_{1n})^\top$, we have that

$$\frac{d}{d\tau}y_1(\tau) \sim \frac{1}{12h}\begin{pmatrix} y_{1(n-1)} - 8y_{1n} + 8y_{11} - y_{12} \\ y_{1n} - 8y_{10} + 8y_{12} - y_{13} \\ \vdots \\ y_{1(n-3)} - 8y_{1(n-2)} + 8y_{1n} - y_{10} \\ y_{1(n-2)} - 8y_{1(n-1)} + 8y_{10} - y_{11} \end{pmatrix}.$$

This can be rewritten as

$$\frac{d}{d\tau}y_1(\tau) \sim \begin{pmatrix} 0 & 8 & -1 & 0 & 0 & 0 & \ldots & 0 & 1 & -8 \\ -8 & 0 & 8 & -1 & 0 & 0 & 0 & \ldots & 0 & 1 \\ 1 & -8 & 0 & 8 & -1 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 1 & -8 & 0 & 8 & -1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & \ldots & 0 \\ \vdots & \vdots & & & & & & \vdots & & \vdots \\ 0 & 0 & \ldots & 0 & 1 & -8 & 0 & 8 & -1 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 1 & -8 & 0 & 8 & -1 \\ -1 & 0 & 0 & 0 & \ldots & 0 & 1 & -8 & 0 & 8 \\ 8 & -1 & 0 & 0 & 0 & \ldots & 0 & 1 & -8 & 0 \end{pmatrix} \begin{pmatrix} y_{10} \\ y_{11} \\ y_{12} \\ y_{13} \\ \vdots \\ y_{1(n-3)} \\ y_{1(n-2)} \\ y_{1(n-1)} \\ y_{1n} \end{pmatrix}.$$

We will let $\boldsymbol{D_{d\tau}}$ be the $(n+1)$ by $(n+1)$ matrix above . Then we can approximate the derivative of $\boldsymbol{Y}(\tau)$ with $\overline{\boldsymbol{D_{d\tau}}}\overline{\boldsymbol{Y}}$ where $\overline{\boldsymbol{D_{d\tau}}}$ is the following block diagonal matrix,

$$\overline{\boldsymbol{D_{d\tau}}} = \begin{pmatrix} \boldsymbol{D_{d\tau}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D_{d\tau}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{D_{d\tau}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{D_{d\tau}} \end{pmatrix}. \tag{3.56}$$

In summary, we have just discretized the derivative operator and now have that $\boldsymbol{Y}' \sim \overline{\boldsymbol{D_{d\tau}}}\overline{\boldsymbol{Y}}$.

Next we will discretize the advancement operator + which is used in (3.54). Recall that $\boldsymbol{Y}_+(\tau) = \boldsymbol{Y}(\tau + T)$. We will discretize the advancement operator so that $\boldsymbol{Y}_+$ can be approximated by some matrix multiplied by $\overline{\boldsymbol{Y}}$. We will now find a suitable matrix. We will first do the discretization of the advancement operator applied to just the function $y_1(\tau)$. Then we can extend the results to the case of the

advancement operator applied to $\boldsymbol{Y}(\tau)$ using a block diagonal matrix just as we did for the derivative operator in (3.56).

The problem is the following. We have a periodic function $y_1(\tau)$ defined on $[0, P)$. The continuous function y_1 is approximated by the discretized vector $(y_{10}, y_{11}, \ldots, y_{1n})^\top$. We would like to approximate $y_1(\tau + T)$ with one of the discretized value y_{1i} where $i = 0, 1, \ldots, n$. First, it may be that $\tau + T \geq P$. This is easily resolved because $y_1(\tau)$ is periodic with period P. If y_1 is required outside the range $[0, P)$ we can always find an equivalent value of y_1 evaluated at some other time in the range $[0, P)$.

Now suppose we want to evaluate $y_1(\tau+T)$ where $\tau = kh$ for some $k \in \{0, 1, 2, \ldots, n\}$. Since y_1 is periodic we will assume that we have shifted time so that $\tau + T < P$. The issue is that $kh + T$ may not have a corresponding mesh point in $\{0, h, \ldots, nh\}$. This is most likely the case since T/h probably will not be a whole number. This means that that $y_1(kh + T)$ can not be simply approximated by one of the values in the set $\{y_0, y_1, \ldots, y_n\}$. Therefore the problem becomes one of expressing the quantity $y_1(kh + T)$ as a linear combination of some of the known values within the set $\{y_0, y_1, \ldots, y_n\}$.

We assume that $y_1(kh + T) = y_1((j + \mu)h)$ for some $j \in \{0, 1, 2, \ldots, n\}$. Here μ satisfies $0 \leq \mu \leq 1$ and is the fraction of the distance between between j and $j + 1$. We can find the value of μ by evaluating $\mu = \frac{T}{h} - \lfloor \frac{T}{h} \rfloor$. Here we have used the floor function defined as $\lfloor x \rfloor = \max\{m \in \mathbb{Z} | m < x\}$.

As a first approximation to $y_1((j + \mu)h)$ we could use y_{1j}. To get a better approximation we could use a weighted average with y_{1j} and $y_{1(j+1)}$ which would lead to $y_1((j + \mu)h) \approx y_{1j} + \mu(y_{1(j+1)} - y_{1j})$. As discussed previously we would like a higher order approximation. We will construct an interpolating polynomial through five mesh points to approximate $y_1((j+\mu)h)$. The problem is depicted in Figure 3.6.

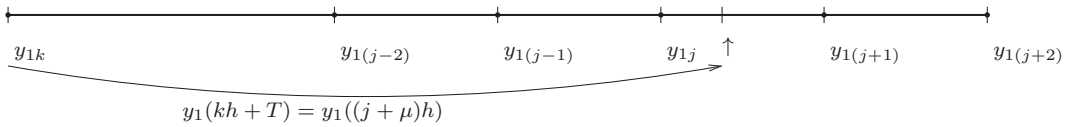


Figure 3.6: To approximate $y_1(kh + T) = y_1((j + \mu)h)$ we use interpolation to express the approximation of $y_1((j + \mu)h)$ as a linear combination of known mesh points around it.

We use an interpolating polynomial defined as follows

$$\begin{aligned} P(x) &= p_0 + p_1 x + p_2 x^2 + p_3 x^3 + p_4 x^4 \\ P(-2) &= y_{1(j-2)}, \quad P(-1) = y_{1(j-1)}, \quad P(0) = y_{1(j)}, \\ P(1) &= y_{1(j+1)}, \quad P(2) = y_{1(j+2)}. \end{aligned}$$

Solving for the coefficients of $P(x)$ leads to

$$\begin{aligned} p_0 &= y_{1j} \\ p_1 &= -\frac{2}{3}y_{1(j-1)} + \frac{1}{12}y_{1(j-2)} + \frac{2}{3}y_{1(j+1)} - \frac{1}{12}y_{1(j+2)} \\ p_2 &= -\frac{5}{4}y_{1j} + \frac{2}{3}y_{1(j-1)} - \frac{1}{24}y_{1(j-2)} + \frac{2}{3}y_{1(j+1)} - \frac{1}{24}y_{1(j+2)} \\ p_3 &= \frac{1}{6}y_{1(j-1)} - \frac{1}{12}y_{1(j-2)} - \frac{1}{6}y_{1(j+1)} + \frac{1}{12}y_{1(j+2)} \\ p_4 &= \frac{1}{4}y_{1j} - \frac{1}{6}y_{1(j-1)} + \frac{1}{24}y_{1(j-2)} - \frac{1}{6}y_{1(j+1)} + \frac{1}{24}y_{1(j+2)}. \end{aligned}$$

Therefore we can approximate $y_1((j+\mu)h)$ with $P(\mu)$.

Next we need to approximate $y_1(\tau + T)$ by approximating $y_1(kh + T)$ for all k in $\{0, 1, 2, \ldots, n\}$. We can do this by multiplying the vector $(y_{10}, y_{11}, \ldots, y_{1n})^\top$ by a matrix of the form

$$\boldsymbol{P} = \boldsymbol{p}_0 + \boldsymbol{p}_1\mu + \boldsymbol{p}_2\mu^2 + \boldsymbol{p}_3\mu^3 + \boldsymbol{p}_4\mu^4, \tag{3.57}$$

where μ is still defined as $\mu = \frac{T}{h} - \lfloor \frac{T}{h} \rfloor$. The coefficients are now the matrices

$$\begin{aligned} \boldsymbol{p}_0 &= \boldsymbol{R}_N \\ \boldsymbol{p}_1 &= -\frac{2}{3}\boldsymbol{R}_{N-1} + \frac{1}{12}\boldsymbol{R}_{N-2} + \frac{2}{3}\boldsymbol{R}_{N+1} - \frac{1}{12}\boldsymbol{R}_{N+2} \\ \boldsymbol{p}_2 &= -\frac{5}{4}\boldsymbol{R}_N + \frac{2}{3}\boldsymbol{R}_{N-1} - \frac{1}{24}\boldsymbol{R}_{N-2} + \frac{2}{3}\boldsymbol{R}_{N+1} - \frac{1}{24}\boldsymbol{R}_{N+2} \\ \boldsymbol{p}_3 &= \frac{1}{6}\boldsymbol{R}_{N-1} - \frac{1}{12}\boldsymbol{R}_{N-2} - \frac{1}{6}\boldsymbol{R}_{N+1} + \frac{1}{12}\boldsymbol{R}_{N+2} \\ \boldsymbol{p}_4 &= \frac{1}{4}\boldsymbol{R}_N - \frac{1}{6}\boldsymbol{R}_{N-1} + \frac{1}{24}\boldsymbol{R}_{N-2} - \frac{1}{6}\boldsymbol{R}_{N+1} + \frac{1}{24}\boldsymbol{R}_{N+2}. \end{aligned} \tag{3.58}$$

Here, the number N is defined as $N = \lfloor \frac{T}{h} \rfloor$. The matrix $\boldsymbol{R}_M$ is a permutation matrix of size $(n+1) \times (n+1)$, which when multiplied by the vector $(y_{10}, y_{11}, \ldots, y_{1n})^\top$, shifts every element in that vector back M spaces. For example,

$$\boldsymbol{R}_2 \cdot (y_{10}, y_{11}, y_{12}, \ldots, y_{1(n-2)}, y_{1(n-1)}, y_{1n})^\top = (y_{12}, y_{13}, y_{14}, \ldots, y_{1n}, y_{10}, y_{11})^\top.$$

Note again that we are using periodicity and this is the reason we can use permutation matrices in the first place. The matrix $\boldsymbol{R}_M$ has the form

$$\boldsymbol{R}_M = \begin{pmatrix} \mathbf{0} & \boldsymbol{I}_{n+1-M} \\ \boldsymbol{I}_M & \mathbf{0} \end{pmatrix}, \tag{3.59}$$

where $\boldsymbol{I}_M$ is an identity matrix of size $M \times M$ and so on.

Now that we have established $y_1(\tau + T) \sim \boldsymbol{P}(y_{10}, y_{11}, \ldots, y_{1n})^\top$, all that is left is to approximate $\boldsymbol{Y}_+(\tau)$. To approximate $\boldsymbol{Y}_+(\tau)$ we multiply $\overline{\boldsymbol{Y}}$ by the block diagonal matrix

$$\overline{\boldsymbol{P}} = \begin{pmatrix} \boldsymbol{P} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{P} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{P} \end{pmatrix}, \tag{3.60}$$

where $\boldsymbol{P}$ is the $(n+1) \times (n+1)$ matrix defined by (3.57), (3.58), and (3.59). Finally, we now have $\boldsymbol{Y}_+ \sim \overline{\boldsymbol{P}}\overline{\boldsymbol{Y}}$.

The last terms to discretize in (3.54) are $\boldsymbol{\Omega}$, $\boldsymbol{J_2}$, and $\boldsymbol{K}$. Before we look at a 4×4 time dependent matrix we consider a single function. For example, the function $F_u(t)$ in the expression $F_u(t)y_1(t)$ could be discretized as

$$F_u(t)y_1(t) \sim \begin{pmatrix} f_u(0) & 0 & \ldots & 0 \\ 0 & f_u(h) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & f_u(nh) \end{pmatrix} \begin{pmatrix} y_{10} \\ y_{11} \\ \vdots \\ y_{1n} \end{pmatrix}.$$

For an arbitrary 4×4 time dependent matrix,

$$\boldsymbol{A}(t) = \begin{pmatrix} a_{11}(t) & a_{12}(t) & a_{13}(t) & a_{14}(t) \\ a_{21}(t) & a_{22}(t) & a_{23}(t) & a_{24}(t) \\ a_{31}(t) & a_{32}(t) & a_{33}(t) & a_{34}(t) \\ a_{41}(t) & a_{42}(t) & a_{43}(t) & a_{44}(t) \end{pmatrix},$$

we can discretize $\boldsymbol{A}(t)$ with the matrix $\overline{\boldsymbol{A}}$.

Here, $\overline{\boldsymbol{A}}$ is the $4(n+1) \times 4(n+1)$ matrix obtained from replacing each function a_{ij} in $\boldsymbol{A}(t)$, where $i = 1, 2, 3, 4$ and $j = 1, 2, 3, 4$, with the $(n+1) \times (n+1)$ diagonal

matrix

$$\begin{pmatrix} a_{ij}(0) & 0 & \dots & 0 \\ 0 & a_{ij}(h) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{ij}(nh) \end{pmatrix}.$$

This is the process we use to discretize $\boldsymbol{J_2}$, $\boldsymbol{\Omega}$, and $\boldsymbol{K}$ to get $\overline{\boldsymbol{J_2}}$, $\overline{\boldsymbol{\Omega}}$, and $\overline{\boldsymbol{K}}$.

Now we have everything in place to completely discretize the operator $\mathcal{L}^*$. We can approximate the continuos adjoint operator $\mathcal{L}^*$ in (3.54) with its discretized version

$$\overline{\boldsymbol{L}} \equiv -\overline{\boldsymbol{\Omega}}\,\overline{\boldsymbol{D_{d\tau}}} - \overline{\boldsymbol{J_2}}^{\top}\overline{\boldsymbol{P}} + \overline{\boldsymbol{K}}. \tag{3.61}$$

The kernel of the matrix $\overline{\boldsymbol{L}}$ converges to the kernel of $\mathcal{L}^*$ in the limit as $n \to \infty$. Also note that kernel of $\overline{\boldsymbol{L}}^{\top}$ should converge to the kernel of $\mathcal{L}$. A good way to check if the numerical construction of $\overline{\boldsymbol{L}}$ is correct is to see if the kernel of $\overline{\boldsymbol{L}}^{\top}$ converges to $\left(u_0', v_0', c_1', c_2'\right)^{\top}$ which we know is in the kernel of $\mathcal{L}$. The code which creates the matrix $\overline{\boldsymbol{L}}$ can be found in the Appendix in §A.5.

3.6.2 Revisiting the Hopf Bifurcation in §3.5.2

In this section we evaluate ω_2 for the Hopf bifurcation example in §3.5.2. It was this example that showed the secular growth in the asymptotic approximations of (3.2). We use the same functions F and G as well as the same parameters that were used in §3.5.2. The delay is $T = 2$ and therefore there is a stable periodic orbit.

We first use the method of steps described in §3.4.3 to solve (3.46) and find periodic functions u_0, v_0, c_1, and c_2. We first choose constant history for (3.46). Initially the solutions will not be periodic but as time evolves the solutions converge to a stable periodic solution. Once a periodic solution is reached we spline the solution on the last delay interval and then use this as history in another simulation of (3.46). In this second simulation there is no initial transient in the solution and the numerical solution is periodic for all time.

The next step is to compute the period of the solution just found. In this example the period of the solutions to (3.46) is found to be $P = 13.4711$. Once the period is obtained the solutions u_0, v_0, c_1, and c_2 are stored for one period, $\tau \in [0, P]$. These solutions are required in creating the matrix $\overline{\boldsymbol{J_2}}$. They are also needed in

the solvability condition for ω_2 in (3.53). The history as well as one period of the solutions to (3.46) is plotted in Figure 3.7.

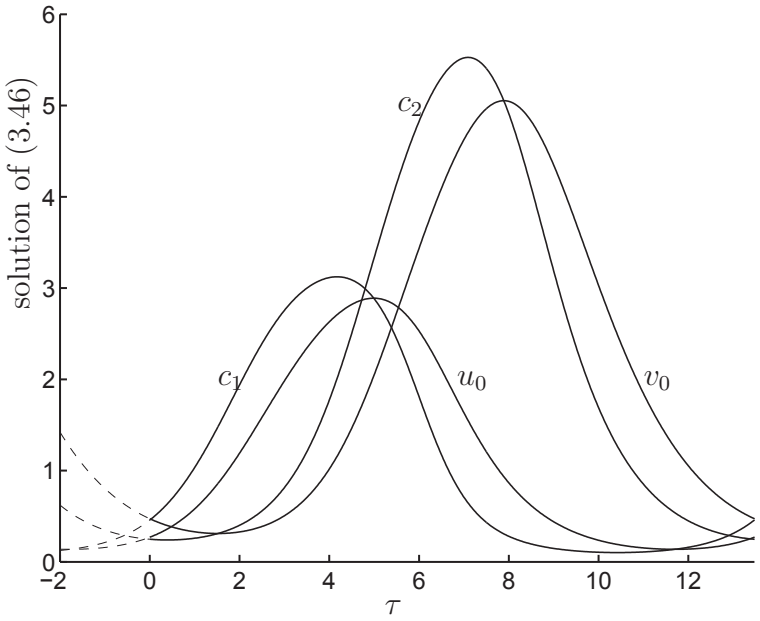


Figure 3.7: The plot of the numerical solutions (solid line) of (3.46) for one period, $\tau \in [0, 13.4711]$. The history (dotted line) is chosen so that there is no initial transient and the solution is periodic for all time. The method of steps, as described in §3.4.1 was used with an absolute and relative tolerance of 1e-9.

In §3.6.1 we formed the matrix $\overline{\boldsymbol{L}}$. We create the matrix $\overline{\boldsymbol{L}}$ in Matlab and then use the *eigs* command to find the zero eigenvalue and its eigenvector. This eigenvector is the vector defined in (3.55). It is split up into four parts. The four parts are the discretized versions of the functions y_1, y_2, y_3, and y_4, which satisfy $\mathcal{L}^*(y_1, y_2, y_3, y_4)^\top = 0$. The discrete vectors can be turned into continuous functions by using the *spline* command in Matlab. The functions y_1, y_2 ,y_3, and y_4 are plotted in Figure 3.8.

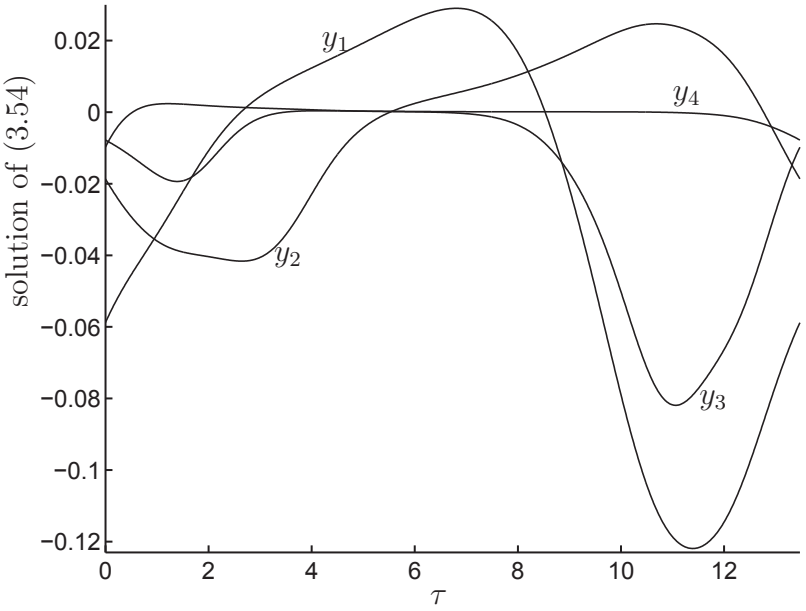


Figure 3.8: A plot of the numerical solutions of (3.54). The vector $\boldsymbol{Y}$ defined in (3.54) is $\boldsymbol{Y} = (y_1, y_2, y_3, y_4)^\top$ and satisfies $\mathcal{L}^*\boldsymbol{Y} = 0$. The discretized version of $\boldsymbol{Y}$ is $\overline{\boldsymbol{Y}}$ and it is the values of $\overline{\boldsymbol{Y}}$ which are used to make this plot. The vector $\overline{\boldsymbol{Y}}$ is an eigenvector of the matrix $\overline{\boldsymbol{L}}$ defined in (3.61). The functions y_1, y_2, y_3, and y_4 are found by splining their discrete counterparts and are used in the solvability condition for ω_2 in (3.53).

Once the functions y_1, y_2 ,y_3, and y_4 are obtained, they are substituted into the solvability condition (3.53). We solve the integrals in (3.53) by using the *quad* command in Matlab. Here is a table of ω_2 versus n showing the convergence.

Table 3.1: Comparison of ω_2 for different values of n. To solve for ω_2 the formula in (3.53) is used. This was done for the parameter values and functions F and G used in §3.5.2.

n	20	40	60	80	100
ω_2	-0.1855457	-0.1867475	-0.1867995	-0.1868093	-0.1868120
n	120	140	160	180	200
ω_2	-0.1868128	-0.1868134	-0.1868136	-0.1868137	-0.1868138

Now that ω_2 has been calculated, it can be used in the system (3.50). We can use the method of steps to solve (3.50). The numerical output from solving (3.50) is then used in (3.41), which is an asymptotic approximation to the solution of (3.2).

With the correct value of ω_2, the functions χ_1, χ_2, A_1, and A_2 will be periodic. The result is shown in Figure 3.9.

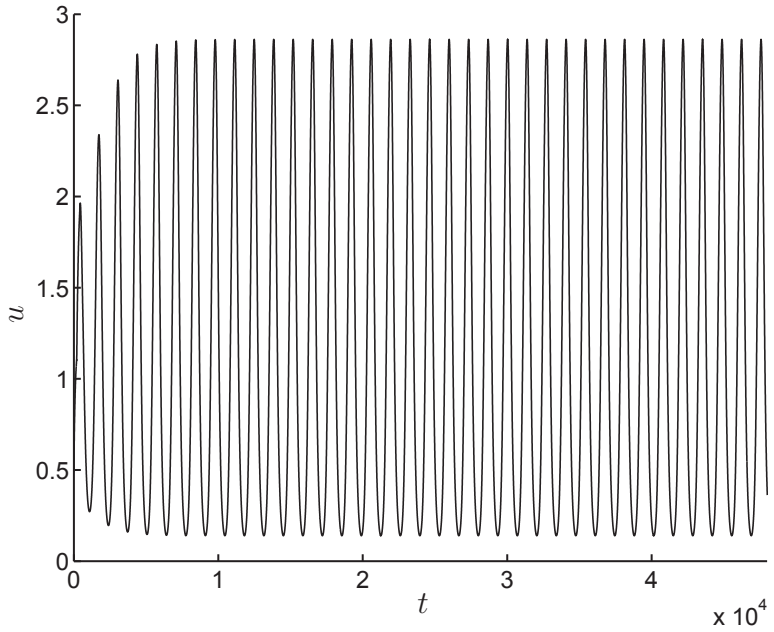


Figure 3.9: This is the plot of $u = u_0 + \varepsilon u_1$ from (3.41). To solve for the terms χ_1 and χ_2 we used the system (3.50) with $\omega_2 = -0.18681$. In this plot there is no secular growth. This is contrasted with the plot in Figure 3.5 where $\omega_2 = 0$. Now the asymptotic solution agrees with the numerical solution of (3.2).

All the Matlab code used here such as the construction of $\overline{\boldsymbol{L}}$, solving the DDAEs, calculating the period, and computing the kernel of $\overline{\boldsymbol{L}}$ can be found in the Appendix in §A.5.

3.6.3 Revisiting the Hopf Bifurcation in §2.5.3

In the previous sections we used the Poincaré-Lindstedt method to eliminate secular growth in the DDAEs in (3.47) in the case of a Hopf bifurcation. We first noticed this secular growth in the delay model. This growth actually occurs in the ODE systems for χ_1 ,χ_2 , and χ_3 from Chapter 2 without delay too. Recall that we did the analysis for a system of two variables but needed a system of three variables to observe a Hopf bifurcation. We did not notice the secular growth in the Hopf bifurcation example in §2.5.3 because we did not solve out far enough in time. Recall that the secular

growth does not occur in the PDE model but only in the ODEs found in (2.50) (extended to a system of three ODEs).

In §2.5.3 we only solved out to time $t = 6000$ and therefore did not observe the secular growth. The asymptotic approximation of $u(x,t)$ from the example in §2.5.3 is plotted in Figure 3.10 further out in time. The secular growth is evident.

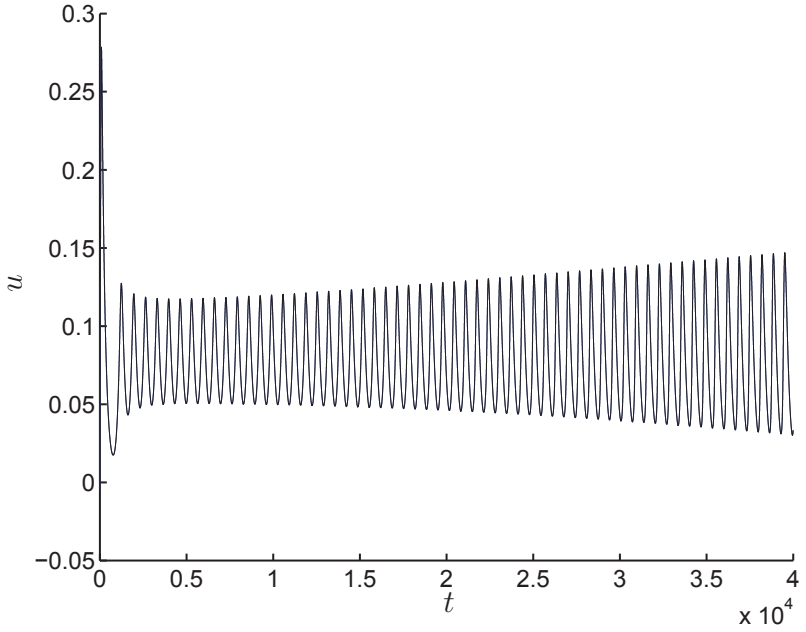


Figure 3.10: This is the plot of u from (2.51) which includes the secular growth coming from the term χ_1. The function χ_1 is determined by the ODEs in (2.50) (extended to a system of three ODEs).

We can repeat the Poincaré-Lindstedt analysis on the PDE model with out delay in (2.3). We will do it here for two variables but it can easily be extended to three variables. We begin by setting the delay to zero in (3.47). Notice that we can isolate for A_1 and A_2 in terms of χ_1 and χ_2 by solving the linear equations in (3.47). This is because $T = 0$ now. This leads to the following system.

$$\begin{pmatrix} \frac{\tau_u}{3}\chi_1 \\ \frac{\tau_v}{3}\chi_2 \end{pmatrix}' = -\begin{pmatrix} \frac{\alpha_1^2}{3} & 0 \\ 0 & \frac{\alpha_2^2}{3} \end{pmatrix}\begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} + \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} - \omega_2 \begin{pmatrix} \frac{\tau_u}{3}u_0 \\ \frac{\tau_v}{3}v_0 \end{pmatrix}'$$

$$\begin{pmatrix} A_1 \\ A_2 \end{pmatrix} = \begin{pmatrix} \frac{B_1}{1-F_u} \\ \frac{B_2}{1-G_v} \end{pmatrix} + \begin{pmatrix} \frac{F_u}{1-F_u} & \frac{F_v}{1-F_u} \\ \frac{G_u}{1-G_v} & \frac{G_v}{1-G_v} \end{pmatrix}\begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix}.$$

Substituting the expressions for A_1 and A_2 into the ODEs for χ_1 and χ_2 results in

$$\begin{pmatrix} \frac{\tau_u}{3}\chi_1 \\ \frac{\tau_v}{3}\chi_2 \end{pmatrix}' + \begin{pmatrix} \frac{\alpha_1^2}{3} - \frac{F_u}{1-F_u} & \frac{F_v}{1-F_u} \\ \frac{G_u}{1-G_v} & \frac{\alpha_2^2}{3} - \frac{G_v}{1-G_v} \end{pmatrix} \begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} = \begin{pmatrix} \frac{B_1}{1-F_u} \\ \frac{B_2}{1-G_v} \end{pmatrix} - \omega_2 \begin{pmatrix} \frac{\tau_u}{3}u_0 \\ \frac{\tau_v}{3}v_0 \end{pmatrix}' \tag{3.62}$$

From the above equation we choose to define the differential operator $\mathcal{L}$ as

$$\mathcal{L}\boldsymbol{Y} \equiv \boldsymbol{\Omega}\boldsymbol{Y}' + \boldsymbol{A}\boldsymbol{Y}.$$

where the matrices $\boldsymbol{\Omega}$ and $\boldsymbol{A}$ are defined as

$$\Omega = \begin{pmatrix} \frac{\tau_u}{3} & 0 \\ 0 & \frac{\tau_v}{3} \end{pmatrix}, \quad \boldsymbol{A} = \begin{pmatrix} \frac{\alpha_1^2}{3} - \frac{F_u}{1-F_u} & \frac{F_v}{1-F_u} \\ \frac{G_u}{1-G_v} & \frac{\alpha_2^2}{3} - \frac{G_v}{1-G_v} \end{pmatrix}.$$

Therefore we can write the ODE system for χ_1 and χ_2 as

$$\mathcal{L}\begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} = \begin{pmatrix} \frac{B_1}{1-F_u} \\ \frac{B_2}{1-G_v} \end{pmatrix} - \omega_2 \begin{pmatrix} \frac{\tau_u}{3}u_0 \\ \frac{\tau_v}{3}v_0 \end{pmatrix}'$$

To find the adjoint operator we repeat the procedure used in §3.6 to obtain

$$\mathcal{L}^*(\boldsymbol{Z}) = -\boldsymbol{\Omega}\boldsymbol{Z}' + \boldsymbol{A}^\top \boldsymbol{Z}.$$

The solvability condition for ω_2 is found the same way as in §3.6. We start with letting $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ be in the kernel of $\mathcal{L}^*$. To eliminate the secular growth in (3.62) we require that

$$\left\langle \begin{pmatrix} \frac{B_1}{1-F_u} - \omega_2 \frac{\tau_u}{3}u_0' \\ \frac{B_2}{1-G_v} - \omega_2 \frac{\tau_v}{3}v_0' \end{pmatrix}, \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right\rangle = 0.$$

Solving for ω_2 yields

$$\omega_2 = \frac{\int_0^P \frac{B_1}{1-F_u}y_1 + \frac{B_2}{1-G_v}y_2 \, d\tau}{\int_0^P \frac{\tau_u}{3}u_0'y_1 + \frac{\tau_v}{3}v_0'y_2 \, d\tau}.$$

We can solve the above equation as well as for y_1 and y_2 using the same techniques described in §3.6.1 and §3.6.2. It is a lot easier to discretize the operator $\mathcal{L}^*$ because it has a simpler form when the delay is zero. Using these techniques on the Hopf bifurcation example in §2.5.3 leads to an ω_2 value of $\omega_2 = -0.392276$. This value of ω_2 eliminates the secular growth in the functions χ_1, χ_2, and χ_3. Here we plot

the same function $u(x,t)$ that was plotted in Figure 3.10 but now there is no secular growth.

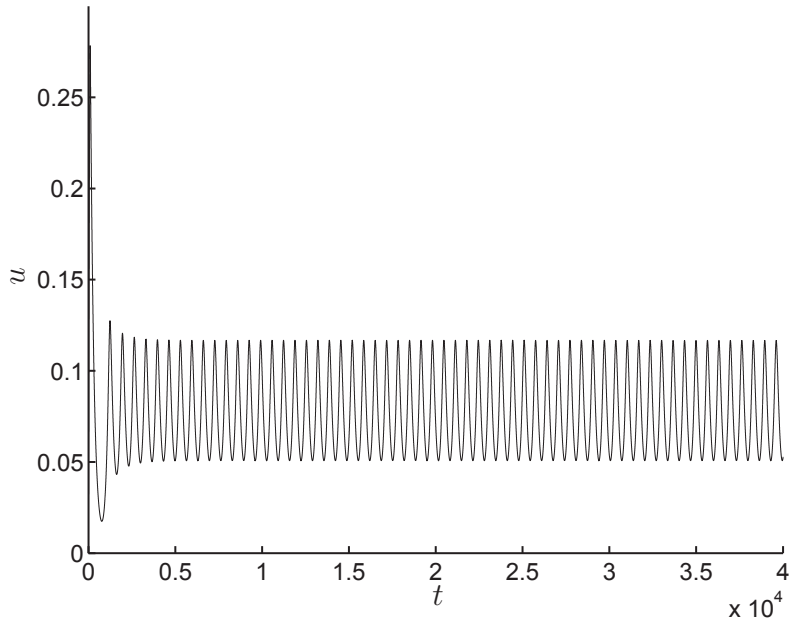


Figure 3.11: This is the plot of u from (2.51). Now the function χ_1 is determined by the ODE system (3.62) (extended to a system of three ODEs) with $\omega_2 = -0.392276$. There is no secular growth.

We leave the discussion of the results in this chapter to the last chapter in this thesis, Chapter 5.

Chapter 4

A Model with Cell Surface Receptors

4.1 Cell Surface Receptors

In the models from previous chapters we only considered the intracellular signalling pathways within the cell. In those models we ignored any initial processes involving the cell surface receptors. In this chapter we begin to look at a simple extension of the models considered previously to include signal initiation at the cell surface.

Cell surface receptors are membrane proteins and play a vital role in the communication between the cell and its exterior. Extracellular signalling molecules act as ligands which bind to the cell surface receptors. The binding of a ligand to its receptor causes a conformational change in the receptor which then initiates a sequence of chemical reactions inside the cell.

Cell surface receptors can be in an active state or an inactive state. One of the factors that affects the state of a receptor is whether it is ligand bound or not. The state of the receptor is directly affected by the ligand bound molecule since ligand bound molecules can cause the receptor to favour one of the two states over the other. Ligand bound receptors are more likely to be active where as receptors that are not ligand bound are more likely to be inactive.

The cell surface receptor proteins are able to diffuse around in the membrane and the spatial arrangement of these receptors is important to cell signalling dynamics. One important feature is when extracellular signalling molecules bind to the receptors which then causes the receptors to aggregate or cluster together [46]. The spatial clustering of cell surface receptors is often required to initiate signal transduction pathways. Such clusterings of cell surface receptors have been observed experimentally through the use of fluorescence microscopy [75].

Clusters of cell surface receptors have been observed in many transmembrane signalling systems. These clusters can have hundreds or thousands of receptors which come together. The clusters can form in specific locations on the cell surface and

this process can often be controlled by the cell. One main consequence of receptor clustering is that the cluster can initiate the signal transduction pathway across the membrane of the cell.

There are different ways in which clusters of membrane receptors form. One way is simply through their protein-protein interactions. In this case, if the receptors occupy the same region, they can cluster through protein-protein interactions such as hydrogen bonds. Even though interactions alone between receptors can cause them to cluster, the cell has no control over this process. Moreover, the size of the clusters and whether or not they form is just a function of the number of receptors [25].

Another way in which receptors can cluster is through adapter proteins within the cytosol. The adapter proteins bind to the part of the cell surface receptors which are in the cytosol. Once the adapter proteins attach to the receptors, it affects how the receptors interact with other receptors. The coupling between two receptors which are both bound by protein adapters can be stronger then it would be to if just one of them or neither was bound. The adapter proteins are relatively large and it is the interactions between the bound protein adapters that causes the clustering of the receptors as opposed to the interactions between the receptors themselves. Since the concentration of adaptor proteins within the cytosol can change rapidly, the ratio of adaptor proteins to isolated receptors acts as a parameter which controls whether or not the receptors cluster together. When the ratio of adapter proteins to receptors increases past a threshold value, more receptors cluster together. The cell can thus control the degree of clustering and can cause clusters to form in specific regions on the cell surface.

Not only can clusters form because of internal stimuli within the cell, such as adapter proteins, they can also form because of external stimuli. An example of this is ligand induced clustering. Ligand molecules are usually small and can bind to the extracellular part of the cell surface receptor. Ligand molecules which are bound to extracellular parts of receptors usually do not interact with each other directly. However, the ligand molecules can affect whether or not receptors cluster by the effect that the ligands have on individual receptors. Often receptors can be in two different conformational states, active or inactive. When a ligand is bound to

a receptor it can affect which state the receptor is in.

When several receptors occupy the same region, the interactions between them depend on their respective states and conformation. Certain energy states will cause the receptors to cluster together. Ligand induced clustering is when the ligand bound receptors change states and begin to cluster with neighbouring receptors. In a cluster of receptors, some receptors will be ligand bound and others will not. Moreover, the conformational sate of a receptor is not only affected by whether it is ligand bound, but also by the state of adjacent receptors within its cluster. Therefore the binding of a ligand with a receptor can affect the activity of other receptors in the corresponding cluster which are not ligand bound [25]. This is known as conformational spread, in which the activity of receptors is spread across the cluster [12].

Since the cell surface receptors can diffuse freely and interact with other receptors, the clusters can form dynamically in time. Once formed, the clusters can also diffuse and sometimes do more slowly after grouping [25]. Through conformational spread, the dynamic clustering of receptors can actually increase the sensitivity and responsiveness of the signalling pathway. Conformational spread is an example of how the forming of clusters of cell surface receptors can enhance signalling responses.

The initiation of the signal transduction pathway occurs at locations where receptors cluster together. Upon entering the cell, the signal travels downstream through cascading reactions between intracellular signalling proteins. The cascading reactions, which involve positive and negative feedback loops, are often localized in space at cellular compartments. An example of a simple efficient signalling pathway would be one with a large cluster followed by a signalling compartment close by. Therefore the signalling molecules could enter the cell and diffuse to the first nearby cellular site and then on to subsequent sites. In this simple pathway everything is well organized and arranged in an optimal sequence. Such a pathway is illustrated in Figure 4.1.

Cellular signalling pathways are very complex. It is unlikely that cellular sites within a signalling pathway are organized along a direct line between the membrane and the final destination of the signal, as depicted in Figure 4.1. Also, the signalling compartments may not be in close approximation with one another. Over longer distances, diffusion is sometimes not enough to transport the signal across the cell interior, especially in the presence of deactivating enzymes [48]. The cell interior also

has a high level of molecular crowding and this can be a hinderance to intracellular signalling molecules which transport via diffusion.

(a) Efficient signalling pathway

Figure 4.1: An efficient signalling pathway consisting of a large cluster of receptors followed by a chain of intracellular signalling compartments. The first intracellular signalling compartment is close to the cluster where the signal crosses the membrane. The compartments which follow are close together as well, shortening the distance the molecules diffuse over.

One advantage of receptor clustering is that it provides fast spatial confinement of receptors. Since cell surface receptors can cluster in a multiple regions on the cell surface, it can lead to shorter diffusion distances for signalling molecules [19]. For example, if a signal had to travel from one cluster at the pole of a cell to reach an intermediate messenger at the other pole of the cell, the diffusion distance would be long. If there were two clusters though, then it would be possible to have a second cluster closer to the intermediate messenger which the signal could travel from. This would decrease the diffusion distance. Other benefits of receptor clustering are heightened sensitivity, broader dynamic range, and enhanced specificity [24].

Many mathematical and computational models of cell surface receptor dynamics and ligand binding have been considered [57, 17, 24, 105, 96, 16]. Many of these models involve ODEs, PDEs, and Monte-Carlo simulations. In [16], the authors investigated the effects of receptor clustering on cellular receptor ligand binding.

Monte-Carlo algorithms were used to simulate ligand diffusion and binding. Three different spatial configurations for the cell surface receptors were considered. They first considered a model with receptors homogeneously spread out over the cell surface. Next they considered a model with clustering with "over stacked receptors". Here the receptors are very tightly packed together. The third model involved a particle simulation framework where receptors were clustered but not as compactly as the second model. In this case, receptors within a cluster are still close to each other but the presence of a ligand in the vicinity of one receptor does not influence the binding of a ligand with another receptor in the cluster. It was found that the probability of a ligand and receptor encounter decreased as the amount of clustering increased. This is because the membrane has larger zones with no receptors over which the ligand has to diffuse. However, they found that receptor clustering increases the rebinding probability.

Another interesting situation is approximating the time it takes for a ligand to reach a receptor using diffusion. As mentioned above, if receptors are clustered, it may take longer for the ligand to reach its binding destination. In [41] the motion of a receptor on a domain with occasional trappings in and escapes from confinement regions was considered. The motion of the receptor was modelled with free diffusion. One main calculation in the paper was estimating the confinement time for a brownian particle in a bounded planar domain, whose boundary is reflecting, except for a small absorbing arc. There was also a calculation done for scenarios with anchors in the domain which could terminate the motion of the particle. In this case the probability of reaching a small absorbing arc was also estimated. Other calculations with traps and patches on 2D and 3D domains were carried out in [42, 21, 81]. In these papers asymptotic results are obtained for the mean field passage time.

In [30] a computational study of 2D and 3D dynamics of receptor ligand interactions was carried out. PDEs were used to develop spatiotemporal models that showed trafficking dynamics of ligands, cell surface components, and intracellular signalling molecules. The model consisted of two bulk domains (the extracellular volume and the inside of the cell) separated by a common 2D interface (the cell surface). On the cell surface information is exchanged between the exterior and interior of the cell through ligand receptor interactions. The model is very complex, involves many

PDEs, and uses realistic parameter values. The authors used Comsol to carry out different numerical simulations and changed individual parameters to see the effect on the model.

4.2 A Model with Cell Surface Receptors

For the model in this chapter we will assume that the receptors have already clustered together and have reached an equilibrium state. Therefore the clusters are at fixed locations in space during the time dependent simulations. To be consistent with the theme of this thesis, we will use the same kind of geometry, PDEs, and boundary conditions as used in Chapters 2 and 3.

We will use the unit sphere to represent the cell. The main difference now is that we will also include a number of circular patches on the surface of the sphere. These patches are regions on the cell surface which represent clusters of cell surface receptors. On these patches we define flux boundary conditions which model the signal travelling across the cell membrane. In this regard, the signal can now be activated at the surface, but only in specific regions where the patches are located. An example of a sphere with several patches all of the same size is shown in Figure 4.2.

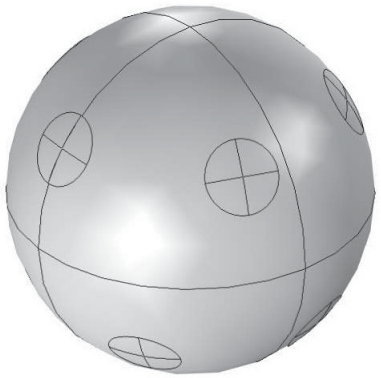


Figure 4.2: A sphere with several circular patches on its surface.

We will start by considering a model where signalling molecules, with concentration $u(x,t)$, are activated at patch locations on the cell surface. Upon activation, these signalling molecules diffuse through the cell interior where they are deactivated.

We use a zero flux boundary condition over the entire sphere surface except on the patch surfaces. A flux boundary condition on the patches models the activation. The model is similar to the models in the previous chapters. Here, instead of having activation at cellular compartments, the activation occurs at the patch locations on the sphere surface. Following the framework of previous chapters, we have the following model for the concentration of the signalling molecules,

$$\begin{aligned}
\tau_u u_t &= \Delta u - \alpha_u^2 u, \quad x \in \Omega_1 \\
\partial_{nx} u &= H(u), \quad x \in \partial\Omega_p \\
\partial_{nx} u &= 0, \quad x \in \partial\Omega_1 \backslash \partial\Omega_p.
\end{aligned}$$

The surface of each patch is denoted as $\partial\Omega_{r_j}$, for $j = 1, 2, \ldots N$, and we define the union of them as $\partial\Omega_p = \cup_{j=1}^{N} \Omega_{r_j}$. Note that we are not necessarily assuming a small decay as we did in the previous chapters.

The geometry for this model consists of the unit sphere with N circular patches on its surface. These patches are just circles projected onto the unit sphere. We will run different simulations while varying the number of patches. For each simulation the patches will collectively cover the same fixed surface area of the sphere. This is because we want to keep the flux or amount of signal entering the cell the same for each simulation. The proportion is P where $0 < P < 1$. Therefore the surface area of each patch is $\frac{4\pi P}{N}$. Now that we have determined the size of the patches we need to choose their locations. To keep consistency between simulations with different numbers of patches, we will use a specific distribution for selecting the centres of the patches. We will use Thomson's problem to do this. Thomson's problem has to do with picking N points on a sphere that are uniformly distributed.

In Thomson's problem [99, 56] the N points are considered as electrons which repel one another by some sort of force law, such as an inverse square force law. The objective is to arrange the N points on the sphere surface so that a stable equilibrium configuration is reached. Saying it another way, the goal of Thomson's problem is to find the minimum energy configuration of N charged electrons constrained to the surface of the unit sphere. J.J. Thomson studied such configurations when considering different models of the atom.

If the sphere is normalized to a radius of 1, and the force is of the inverse-square

type, then there are $N(N-1)/2$ separations $s_{i,j}$ between the N particles. The position of each particle is chosen to minimize the energy

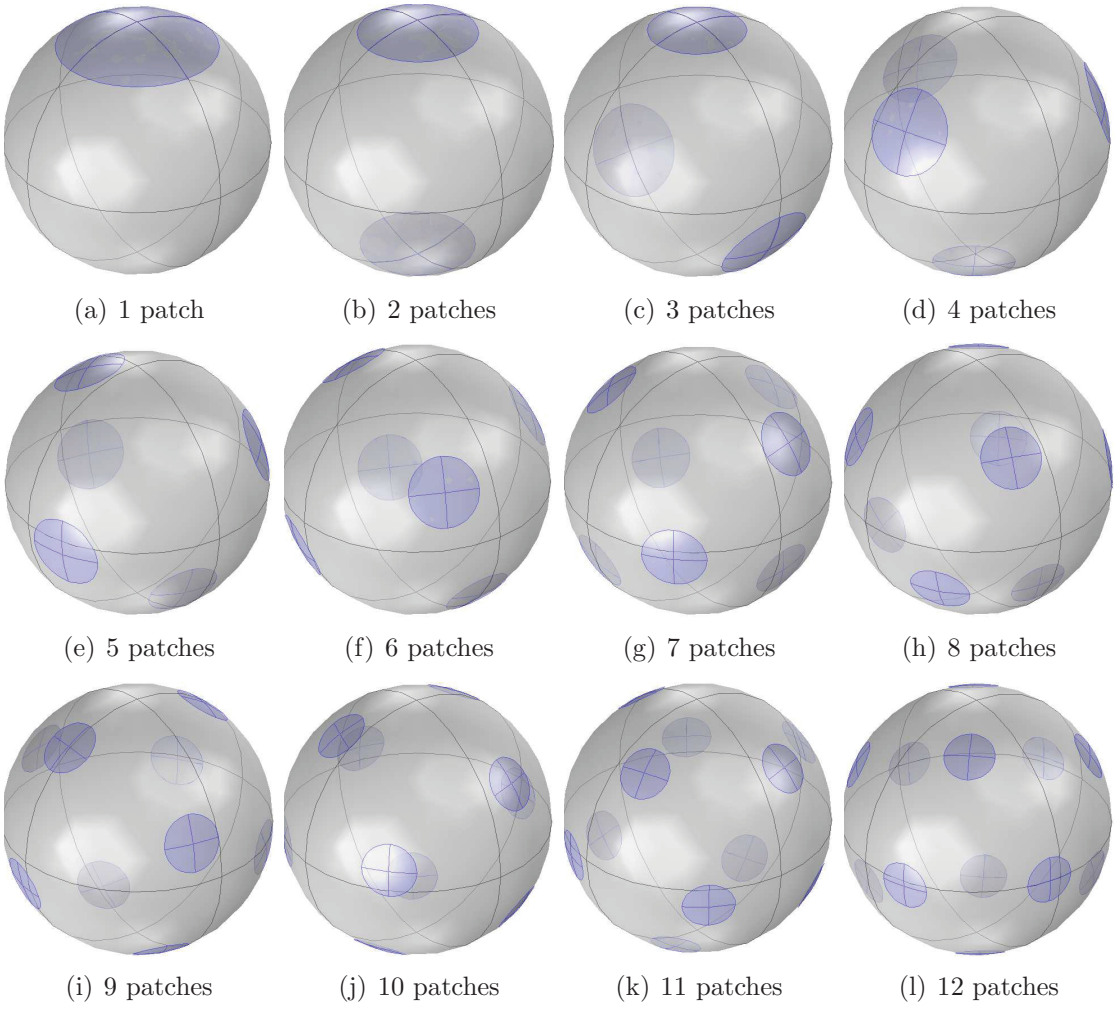
$$E = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{1}{s_{i,j}}.$$

For example, if $N = 0$ then this leaves the sphere with no particles. If $N = 1$ then one particle can be placed anywhere on the sphere such as the north pole. If $N = 2$ then one particle is placed at the north pole and the other at the south pole. If $N = 3$ the three particles are points on a great circle and form an equilateral triangle. For $N = 4$, the four points are vertices of a tetrahedron. Figure 4.3 shows the geometry that we are using in our model for different values of N. These were created in Comsol and show the geometry which is used for the different simulations. In these figures, the patches are covering 10% of the surface area of the unit sphere.

Now we will describe how we create the geometry in the FEM package Comsol. For the simulations in this chapter we used Comsol version 4.4 [3]. First the unit sphere is created, which is a simple operation, and then patches are created on the surface of the sphere. Creating the patches is a little more challenging using the drawing package within Comsol. We wanted to create the patches in a way where it would be easy to define the centre of the patch locations in spherical coordinates. Since the patches are on the surface of the unit sphere we just need to use an azimuth angle θ and a zenith angle ϕ. The angle θ is in the xy plane and is measured from the x-axis with $0 \leq \theta \leq 2\pi$. The angle ϕ is measured from the positive z-axis with $0 \leq \phi \leq \pi$.

First we create a patch in the Comsol graphical user interface (GUI). Then this file can be turned into a script using the LiveLink for MATLAB scripting language. Using the scripting environment, we can edit the code so that we can easily generate any number of patches at specified locations. We have written a script which automates the process of creating the patches. Only the positions and sizes need to be entered in a text file and then the patches are created automatically.

This is the process of creating one patch in the Comsol GUI. First we create an xz work plane. Within that work plane we create a parametric curve of the form $x = \cos t$ and $z = \sin t$ where $\frac{\pi}{2} - \beta \leq t \leq \frac{\pi}{2}$ and $0 < \beta < \pi$. Next, the parametric curve defined in the xz plane is revolved by 360° around the $z-$axis. The final result



(a) 1 patch (b) 2 patches (c) 3 patches (d) 4 patches

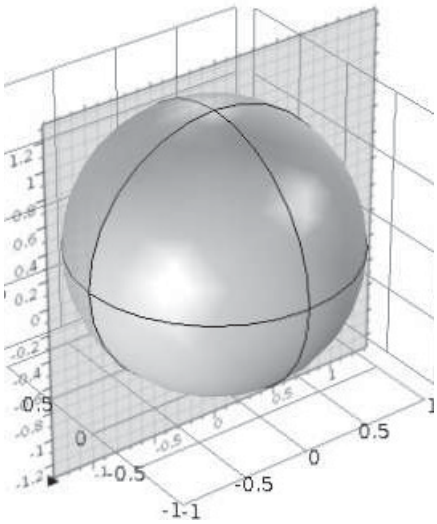
(e) 5 patches (f) 6 patches (g) 7 patches (h) 8 patches

(i) 9 patches (j) 10 patches (k) 11 patches (l) 12 patches

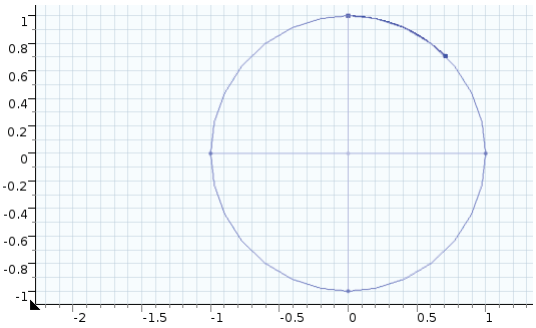
Figure 4.3: The locations for the receptor clusters are chosen to be the points which are solutions to the Thomson problem.

is a patch centred at the point $(0, 0, 1)$. Once this patch is created at the north pole of the sphere it can be rotated about the $y-$axis by an angle ϕ and then rotated about the $z-$axis by an angle θ. The final result is a circular patch centred at the point $(1, \theta, \phi)$, in spherical coordinates. The process of creating a patch at the north pole of the sphere is illustrated in Figure 4.4.

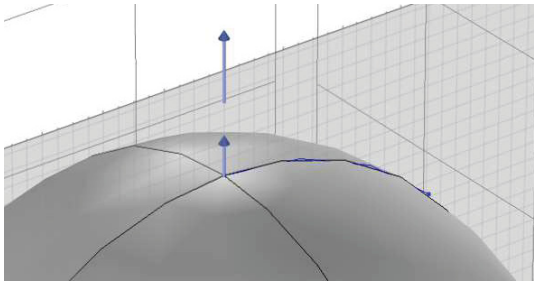
The area of the patch can be calculated by a standard surface area of revolution formula using the parametric curves defined in the xz plane. The parametric curve defined above was revolved around the z-axis. Therefore the area of the patch is



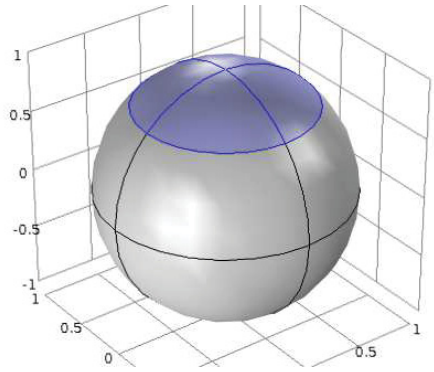
(a) The xz work plane.



(b) Parametric curve in the xz work plane.



(c) Revolving the parametric curve.



(d) Patch centred at the north pole.

Figure 4.4: The steps used in Comsol to create one patch centred at the north pole.

given by the formula

$$\begin{aligned}\text{Patch Area} &= \int_{\frac{\pi}{2}-\beta}^{\frac{\pi}{2}} 2\pi x \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2}\, dt \\ &= \int_{\frac{\pi}{2}-\beta}^{\frac{\pi}{2}} 2\pi \cos(t)\, dt \\ &= 2\pi\left(1 - \cos\beta\right).\end{aligned}$$

To cover a proportion P of the unit spheres surface with N patches of equal size then β must satisfy

$$2\pi\left(1 - \cos\beta\right) = \frac{4\pi P}{N}.$$

Therefore, β is chosen to be

$$\beta = \cos^{-1}\left(\frac{N - 2P}{N}\right).$$

There is one major complication when finalizing the geometry in Comsol with our model. We have described how we have a system where any number of patches can be created quickly using the Comsol scripting environment. The final step in the geometry building process is to union all the patches with the unit sphere to create one physical object, a sphere with N patches on its surface. This is done through a finalize union command in Comsol.

In Comsol, a sphere needs to be divided into surfaces and edges in order to be well defined and work in the geometry environment. Actually, many geometric objects within Comsol need to be divided up into pieces like this. A sphere containing just one continuous surface and one, if any, vertex will not be well defined, nor possible to use in a reasonable way. In Comsol, the sphere is divided into 8 quadrants which are separated by edges. The patches are separated into 4 quadrants which are also separated by edges.

The problem is the following. The final union command will not work, meaning that no simulations can be carried out, if one of the following holds true:

- The edges of the patches are not aligned perfectly with the edges of the sphere.
- The edges of the patches intersect with edges of the sphere.

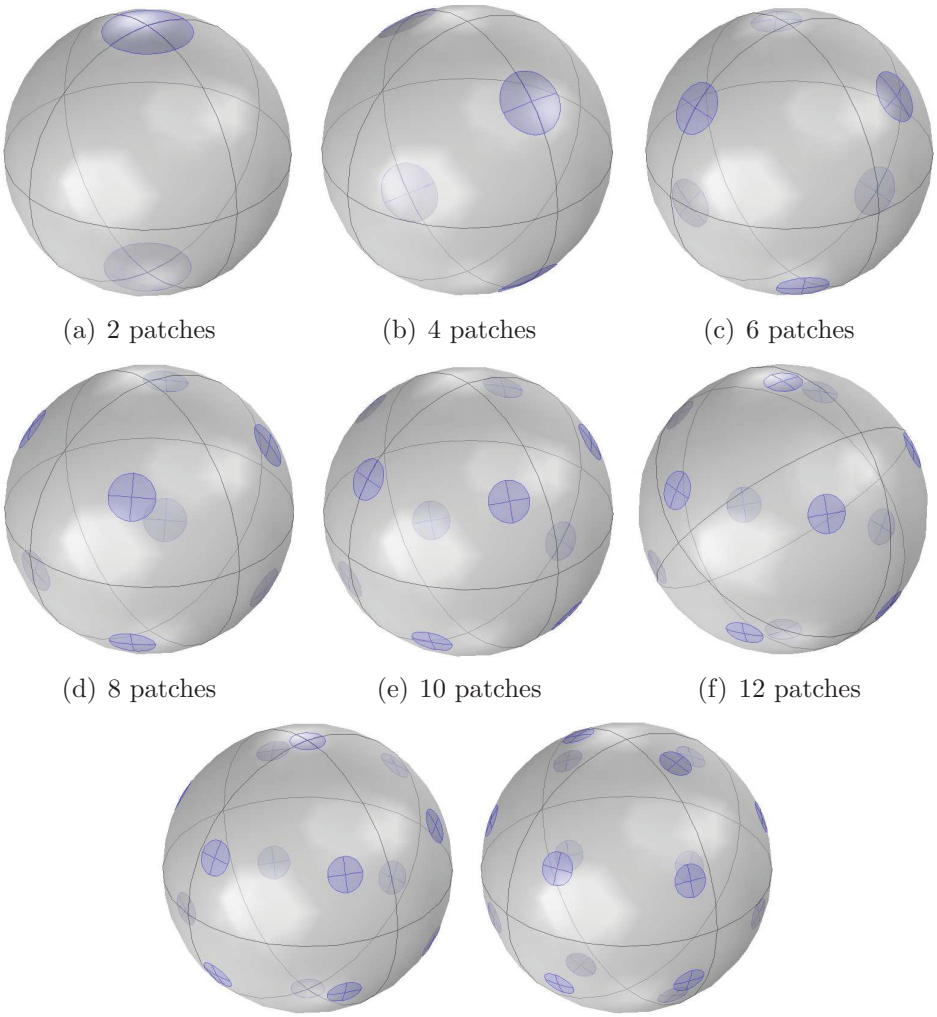
The reason for this is that the default version of Comsol uses something called a Comsol CAD kernel. While the geometry representation of the patches and sphere can be done within the default version of Comsol, the Comsol CAD kernel can not finalize the geometry. A parasolid kernel is required for the geometry in our model because the geometry is complex. This parasolid kernel is only available through the purchase of the CAD Import Module kernel, which is rather expensive.

In summary, if nothing is done with the geometry, we can only do simulations for specific cases. For example, if N is 1, or 2, or 3 then it can be seen from Figure 4.3 that the patch edges align perfectly with the edges of the unit sphere. In these cases we can finalize the geometry. Then the PDEs can be defined and solved using the geometry. If $N = 4$ though, one can see from Figure 4.3 that at least one patch edge intersects with one of the spheres edges. If the geometry is left like that, then no simulations can be carried out. The same goes for almost any other simulation having $N \geq 4$.

The work around to this annoyance was to look at each individual case and manually rotate the sphere through multiple axis so that no patch edges ended up intersecting with any sphere edges. This does not change the energy configuration of course. We went through this process for $N \leq 12$ with the patches covering 5% of the spheres surface. In some cases it was very difficult to find the angles in which to rotate the sphere so that no patch edges would intersect with sphere edges. In those cases (N=9, 10, and 11) we had to physically move 1–3 patches slightly from their desired positions as well as rotate the sphere through several axis. This complication prevented us from running simulations with large values of N and also with larger patch sizes.

One other work around we found was to use a different energy configuration from that of the original Thomson problem. In this variant problem, an additional symmetry constraint is imposed. The goal is to minimize the energy but also have the points symmetric about the xy plane. This problem was treated in [56] and the author gave us the (x, y, z) locations for the patches in this case. For even values of N, if we impose the symmetry in the xy plane it leads to far less complications. For example, we can cover 5% of the sphere surface with patches for $N \in \{2, 4, 6, 8, 10, 12, 14, 16\}$. Only in one of these cases, $N = 12$, did we have to manually rotate the sphere so

that no edges would intersect. In Figure 4.5 we show the patch locations for some even values of N where the points are symmetric about the xy plane.



(a) 2 patches (b) 4 patches (c) 6 patches

(d) 8 patches (e) 10 patches (f) 12 patches

(g) 14 patches (h) 16 patches

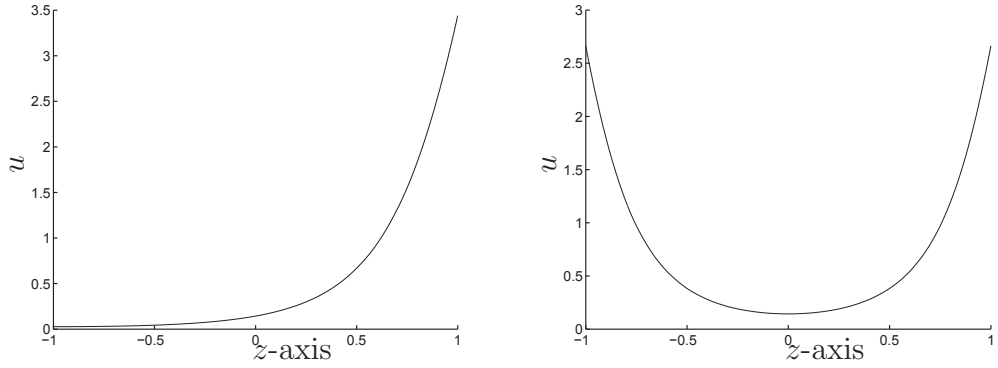
Figure 4.5: The location of the patches using the Thomson's problem with an addition symmetry constraint in the xy plane. Here the patches cover 5% of the spheres surface and only in the case of $N = 12$ did the sphere have to be rotated.

4.3 Example of Receptor Models with Strong and Weak Decay

In this first example we will solve the following model numerically for different values of N,

$$u_t = \Delta u - 6u, \quad x \in \Omega_1$$
$$\partial_{nx} u = 10, \quad x \in \partial\Omega_p$$
$$\partial_{nx} u = 0, \quad x \in \partial\Omega_1 \backslash \partial\Omega_p.$$

We will be using the patch locations shown in Figure 4.3 and the patches will cover 5% of the spheres surface. In this example the decay is chosen so that with just one patch at the north pole, the concentration at the opposing pole is relatively small. In Figure 4.6(a) we plot u at equilibrium along the z-axis with $N = 1$. The concentration is higher near the top of the z-axis where the patch is located. The concentration decays everywhere in the cell and is very low at the opposite side of the cell. If $N = 2$, with one patch at the north pole and one at the south pole, then the concentration profile along the z-axis is higher at both ends of the axis. This is seen in Figure 4.6(b).



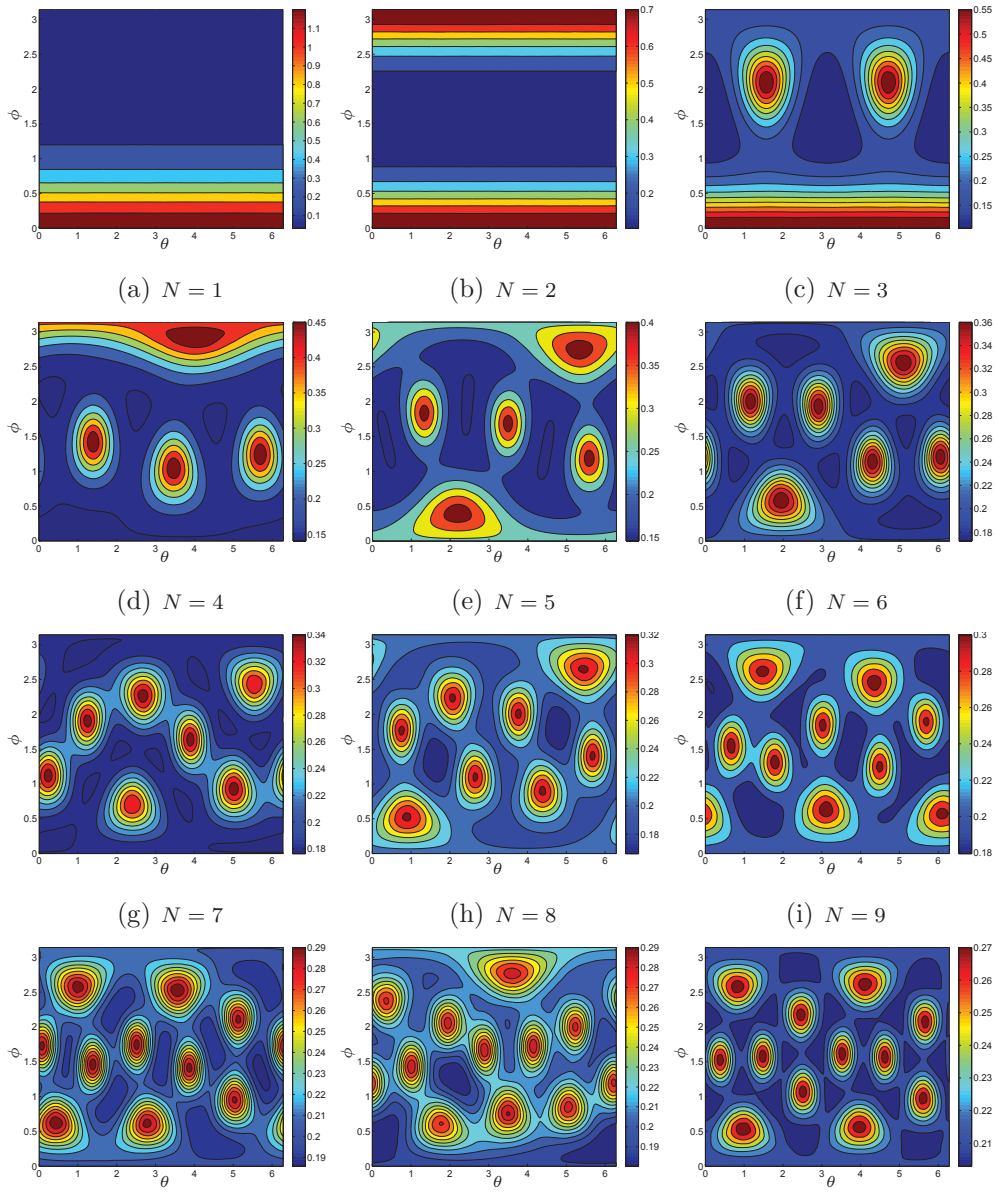
(a) $N = 1$ (b) $N = 2$

Figure 4.6: These plots show the concentration profiles of u along the z-axis for $N = 1$ and $N = 2$.

The concentration $u(x, t)$ is higher in regions which are closer to the patches. In general, as the number of patches is increased, the concentration spreads more evenly across the domain. The maximum concentrations at the patches will be lower for larger N, but the concentrations furthest from the patches will be higher.

In the next series of figures we plot u at equilibrium at numerous points on the surface of a sphere of radius $r = 0.7$. That is we plot u at points of the form $(\theta, \phi, .7)$ in spherical coordinates where $0 \leq \theta \leq 2\pi$ and $0 \leq \phi \leq \pi$. The plots are displayed as contour plots in the θ-ϕ plane (see Figure 4.7). It is clear that a N

increases the maximum concentration decreases. As well, the concentration is more uniformly spread out in the sense that the difference between the lowest and highest concentrations decreases as N increases.



(a) $N = 1$ (b) $N = 2$ (c) $N = 3$

(d) $N = 4$ (e) $N = 5$ (f) $N = 6$

(g) $N = 7$ (h) $N = 8$ (i) $N = 9$

(j) $N = 10$ (k) $N = 11$ (l) $N = 12$

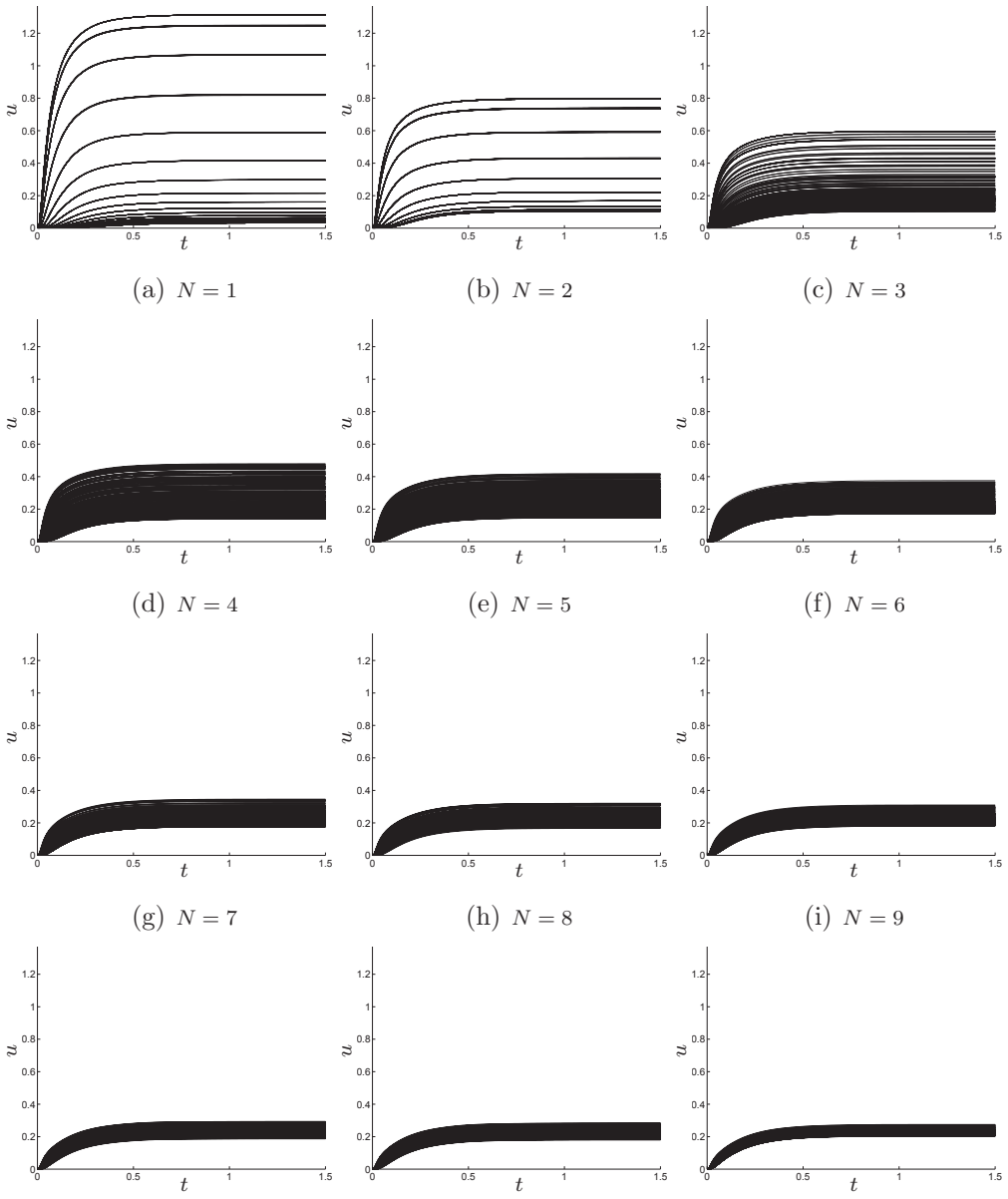
Figure 4.7: These contour plots display the equilibrium value of u at different points in the domain Ω_1. All the points are taken off of a sphere of radius 0.7. The x-axis of each figure is θ and the y-axis of each figure is ϕ.

Thus far we have only looked at the concentration $u(x,t)$ at equilibrium. Now we will consider how the concentration changes in time as the number of patches is

increased. In these next series of figures we plot u as a function of time at numerous points on the surface of a sphere of with radius $r = 0.7$.

From Figure 4.8 we can again see that as N is increased, the concentration becomes more uniform. The difference between the maximum and minimum concentrations is lowered as N increases. As N increases, the concentration decreases at the points which are closer to the patches whereas the concentration at points further from the patches increases. We note that the concentration at the the point (0,0,0) is more or less unaffected as N increases. This makes sense since it is always the same distance from all the patches.

In Figure 4.9, we show the time it takes for $u(x, t)$ to reach 80% of equilibrium for different values of N. There is nothing special about using 80% and it is just a value we chose. This calculation depends on the spatial points chosen in the domain. First we choose a single point in the domain and get u as a function of time at that specific point. We then take that function and subtract 80% of the equilibrium value which is attained at that point. We use the **fsolve** command in MATLAB to calculate the zero crossing of this new function and the solution gives us the time that u takes to reach 80% of equilibrium. We do these calculations for collections of points which are all on a spheres surface of some radius.



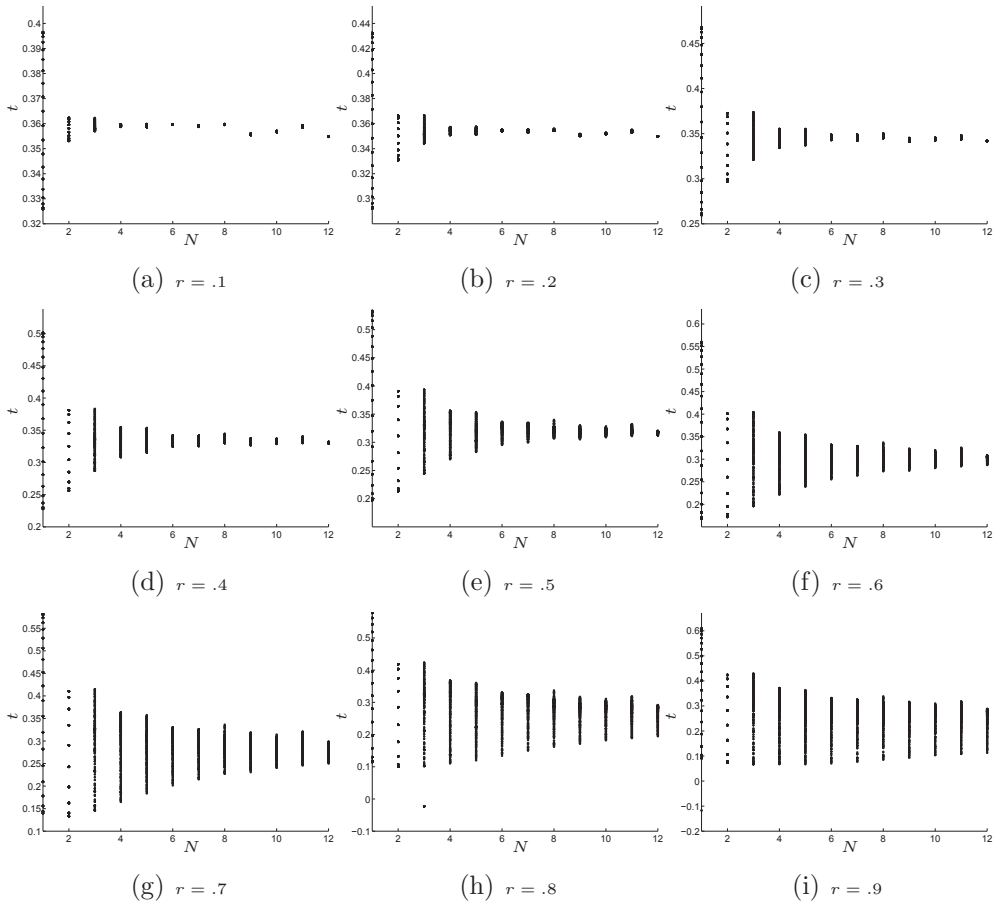
(a) $N = 1$ (b) $N = 2$ (c) $N = 3$

(d) $N = 4$ (e) $N = 5$ (f) $N = 6$

(g) $N = 7$ (h) $N = 8$ (i) $N = 9$

(j) $N = 10$ (k) $N = 11$ (l) $N = 12$

Figure 4.8: Each of these plots displays u as a function of time at numerous points in the domain. The spatial points chosen in each figure are the same and are of the form $(\theta, \phi, 0.7)$ in spherical coordinates.



(a) $r = .1$ (b) $r = .2$ (c) $r = .3$

(d) $r = .4$ (e) $r = .5$ (f) $r = .6$

(g) $r = .7$ (h) $r = .8$ (i) $r = .9$

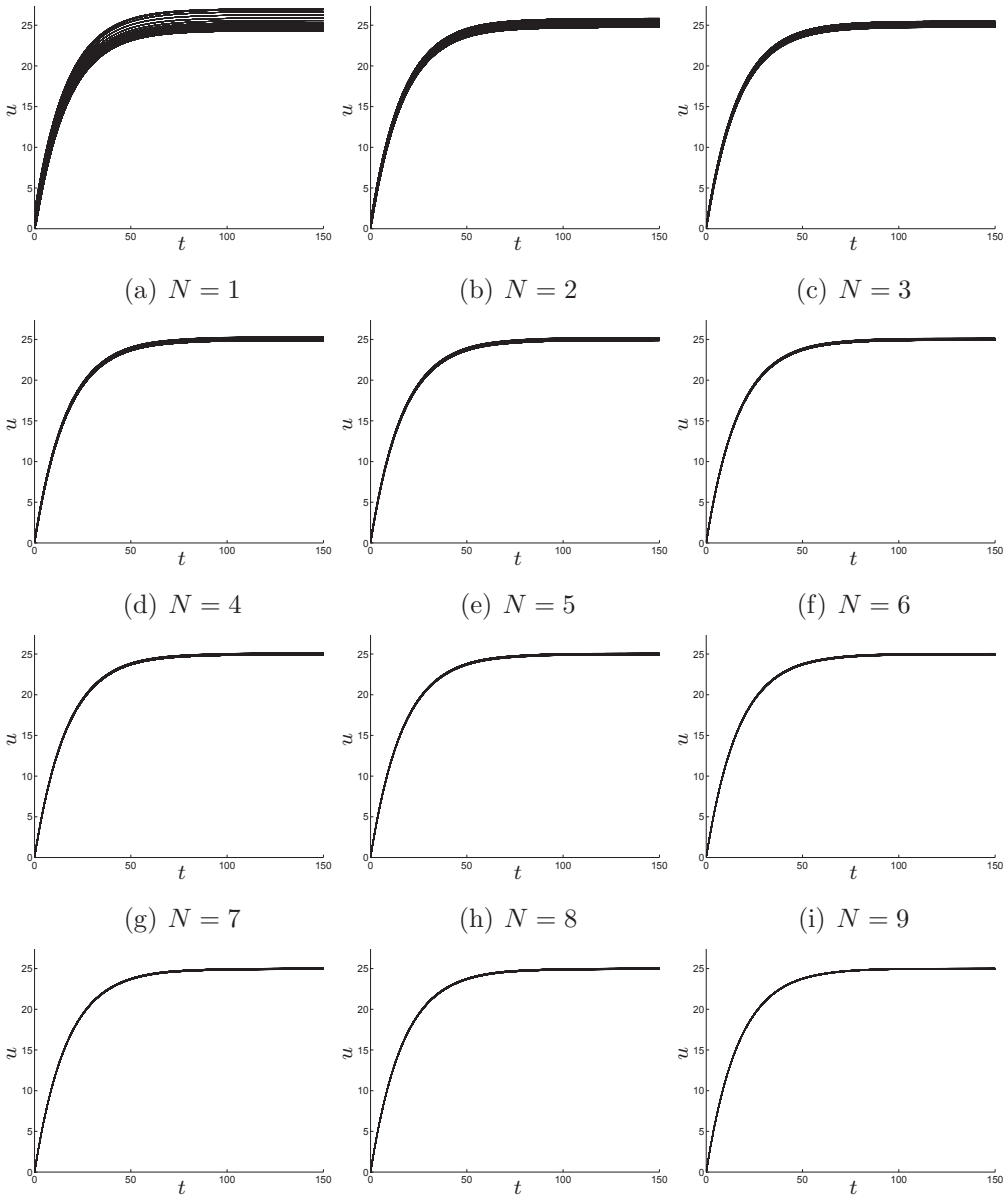
Figure 4.9: These plots show the time it takes for $u(x,t)$ to reach 80% of equilibrium at different points in the domain. Each figure is for a collection of points off the surface of a sphere of radius r.

From the figures in Figure 4.9 we can see that as N increases, the time it takes for u to reach 80% of equilibrium decreases. This effect is more dramatic at points which are further from the cell surface patches. Different points in space reach 80% of equilibrium faster than other points in space. This effect is more dramatic for smaller N. As N increases, the solution is becoming more uniform across the domain. The time at which different points reach 80% of equilibrium becomes more consistent.

Now we repeat these simulations but for the following model where we have changed the decay value to something smaller.

$$\begin{aligned} u_t &= \Delta u - .06u, \quad x \in \Omega_1 \\ \partial_{nx} u &= 10, \quad x \in \partial\Omega_p \\ \partial_{nx} u &= 0, \quad x \in \partial\Omega_1 \backslash \partial\Omega_p \end{aligned}$$

We can see from Figure 4.10 that changing the number of clusters does not have a large effect on the the concentration $u(x,t)$ when the decay is small. Since the decay is small the signalling molecules can diffuse through the domain without being deactivated quickly. For $N > 3$ the concentration of the signalling molecules is basically the same across the domain regardless of the number of clusters. It may be that changing the number of clusters does not have much effect on the signalling pathway when the decay is relatively weak. For completeness, in Figure 4.11, we also include the time it takes for $u(x,t)$ to reach 80% of equilibrium for different values of N in this case of the weak decay.



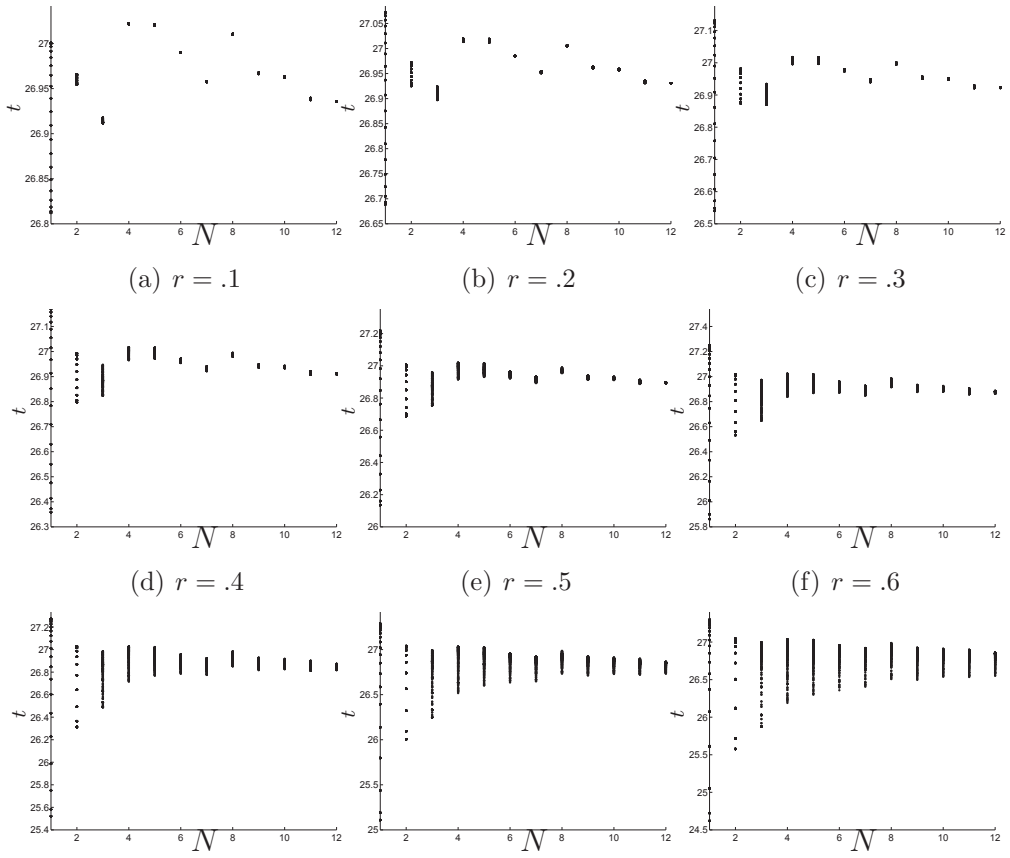
(a) $N = 1$ (b) $N = 2$ (c) $N = 3$

(d) $N = 4$ (e) $N = 5$ (f) $N = 6$

(g) $N = 7$ (h) $N = 8$ (i) $N = 9$

(j) $N = 10$ (k) $N = 11$ (l) $N = 12$

Figure 4.10: Each of these plots displays u as a function of time at numerous points in the domain. The spatial points chosen in each figure are the same and are of the form $(\theta, \phi, 0.7)$ in spherical coordinates.



(a) $r = .1$ (b) $r = .2$ (c) $r = .3$

(d) $r = .4$ (e) $r = .5$ (f) $r = .6$

(g) $r = .7$ (h) $r = .8$ (i) $r = .9$

Figure 4.11: These plots show the time it takes $u(x,t)$ to reach 80% of equilibrium at different points in the domain. Each figure is for a collection of points off the surface of a sphere of radius r.

4.4 Hopf bifurcation in a Model with Patches and Compartments

In this example we consider a signalling model with cellular compartments from Chapter 2 combined with the cell surface receptors introduced in this chapter. The signal transduction pathway can be initiated at the cell surface where the receptors cluster and then travel to the cellular compartments and activate intermediate messengers. The model we consider in this example is

$$
\begin{aligned}
\frac{\partial u_1}{\partial t} &= \Delta_x u_1 - 6u_1, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} u_1 &= \left(1 + \frac{125u_1}{u_1 + 1}\right) \frac{1}{u_4^3 + 1}, \quad x \in \partial\Omega_p \\
\partial_{nx} u_1 &= 0, \quad x \in \partial\Omega_1 \backslash \partial\Omega_p \\
\\
\frac{\partial u_2}{\partial t} &= \Delta_x u_2 - u_2, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} u_2 &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon\partial_{nx} u_2 &= \frac{100u_1^2}{u_1^2 + 1}, \quad x \in \partial\Omega_{\varepsilon_1} \\
\\
\frac{\partial u_3}{\partial t} &= \Delta_x u_3 - u_3, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} u_3 &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon\partial_{nx} u_3 &= \frac{100u_2^2}{u_2^2 + 1}, \quad x \in \partial\Omega_{\varepsilon_2} \\
\\
\frac{\partial u_4}{\partial t} &= \Delta_x u_4 - u_4, \quad x \in \Omega_1 \backslash \Omega_\varepsilon \\
\partial_{nx} u_4 &= 0, \quad x \in \partial\Omega_1 \\
\varepsilon\partial_{nx} u_4 &= \frac{100u_3^2}{u_3^2 + 1}, \quad x \in \partial\Omega_{\varepsilon_3}.
\end{aligned}
\tag{4.1}
$$

The first set of equations for u_1 govern the evolution of the concentration of signalling molecules that are activated at the cell surface receptors. The following equations for u_2, u_3, and u_4 govern the concentrations for the signalling proteins that are activated inside the cell on the surface of subcellular compartments. The location for the three compartments Ω_{ε_1}, Ω_{ε_2}, and Ω_{ε_3} are chosen to be $(-0.5, -0.5, -0.5)$, $(0.6, 0.3, 0.4)$ and $(-0.5, 0, 0.7)$ respectively. There is nothing significant about this particular arrangement. The radius of each compartment is $\varepsilon = 0.1$. The geometry is depicted in Figure 4.12 using 12 patches. Here, Ω_ε is defined to be the union of the three internal compartments.

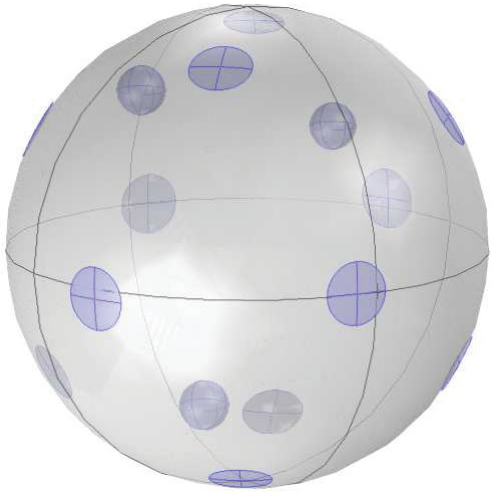
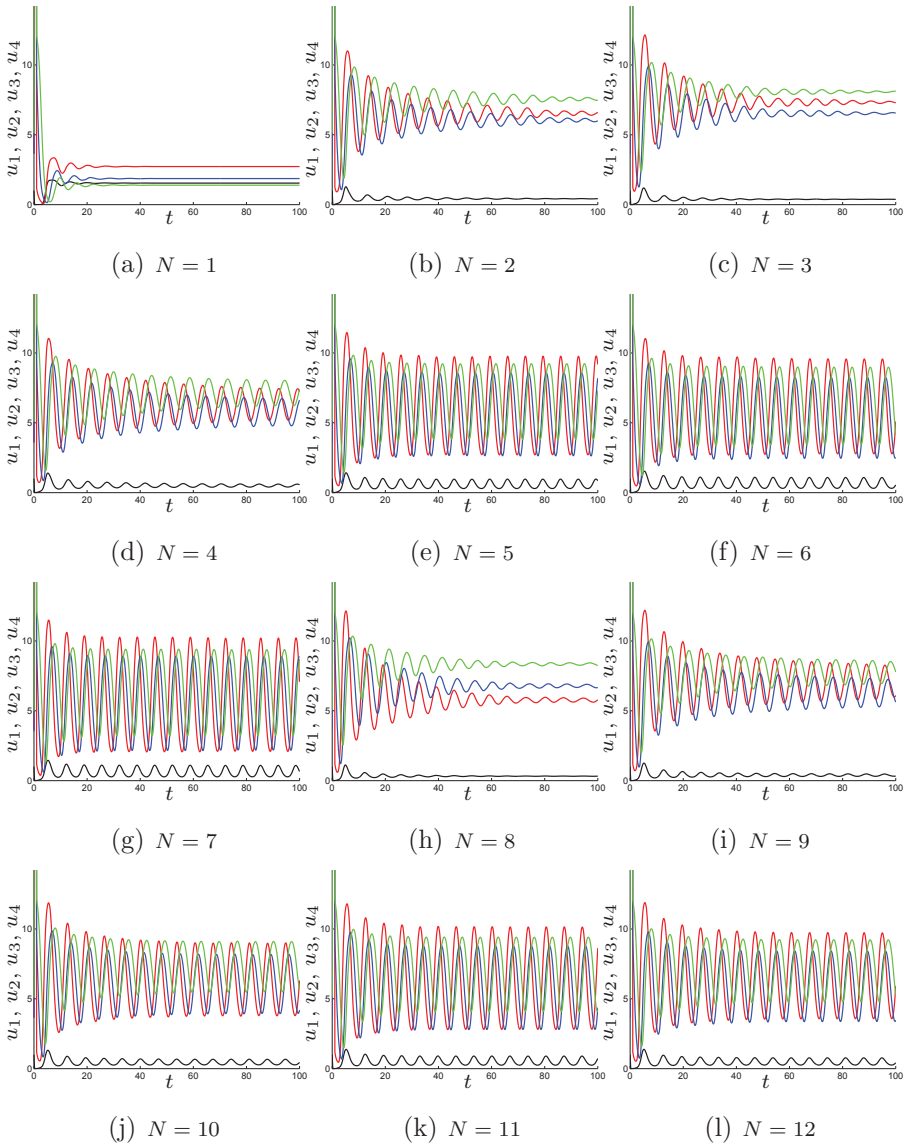


Figure 4.12: For this section we use this geometry with three compartments, Ω_{ε_1}, Ω_{ε_2}, and Ω_{ε_3}, which are located at $(-0.5, -0.5, -0.5)$, $(0.6, 0.3, 0.4)$ and $(-0.5, 0, 0.7)$. The radius of each compartment is $\varepsilon = 0.1$. We will change the number of patches on the surface for different simulations. The patch locations for different values of N are the same as those displayed in Figure 4.3.

Originally we had all the initial conditions set to 0 for all four variables. For some cases the time stepping algorithm had trouble converging. We tried different absolute and relative tolerances but this did not fix the issue in all cases. We believe the issue could be fixed with more tweaking. For simplicity we decided to set non zero constant initial conditions as $[u_1(x,0), u_2(x,0), u_3(x,0), u_4(x,0)] = [1, 3, 2.5, 4]$.



(a) $N = 1$ (b) $N = 2$ (c) $N = 3$

(d) $N = 4$ (e) $N = 5$ (f) $N = 6$

(g) $N = 7$ (h) $N = 8$ (i) $N = 9$

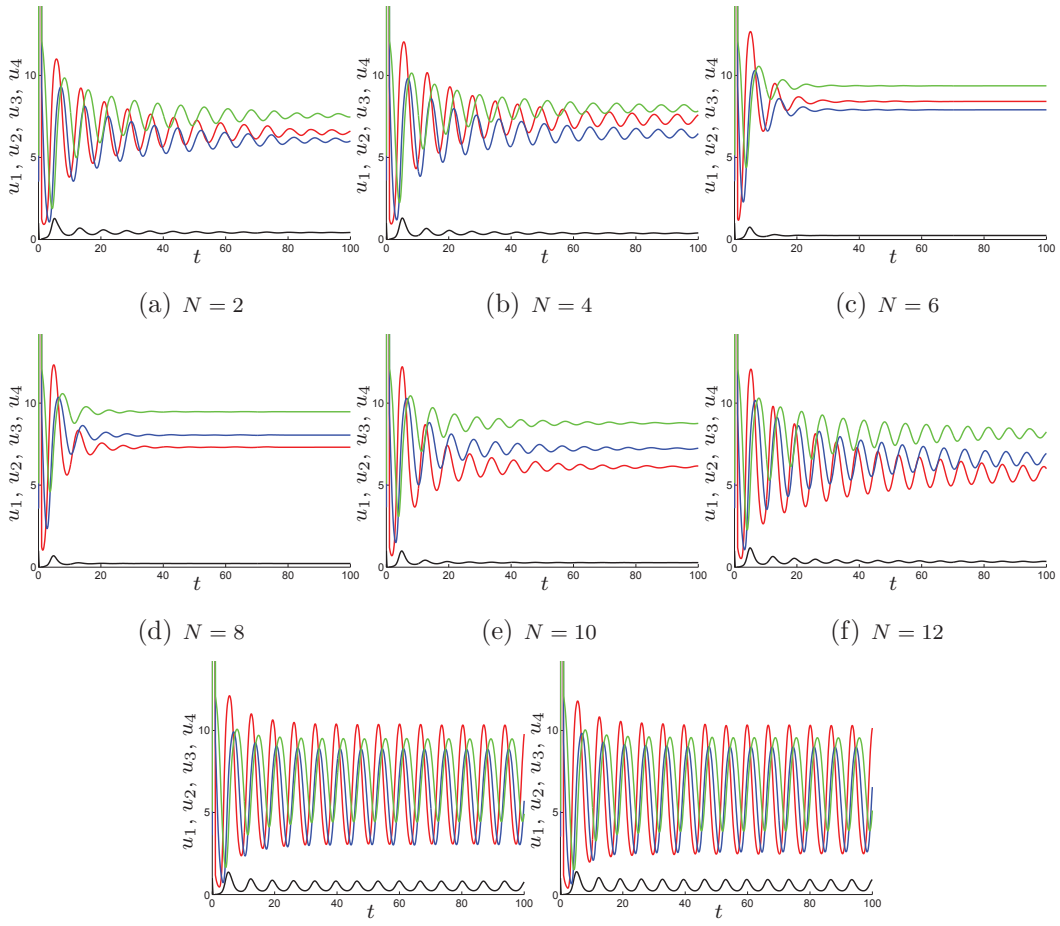
(j) $N = 10$ (k) $N = 11$ (l) $N = 12$

Figure 4.13: In these figures we plot u_1 (black) , u_2 (red), u_3 (blue), and u_4 (green) at specific points in the domain. The model being solved is (4.1). The function u_1 is plotted at the origin and the other functions are plotted at a point on the surface of their respective compartments. Here we see that changing the number of patches can have a dramatic effect on the solution.

It is interesting that sustained oscillations are observed as the number of clusters is increased. Here the bifurcation parameter is the source concentration u_1 coming from the cell surface. The controlling factor for the magnitude of u_1 is the number of clusters. It is also interesting that the sustained oscillations are not present when $N = 8$ (and maybe even for $N = 9$). The problem may be a bit sensitive in this regime. It also may have something to do with the pattern being disrupted when we

had to manually move a few of the compartments in the cases $N = 9, 10, 11$.

Next we will solve the model in (4.1) using the patch geometry from Figure 4.5. In this case the cluster locations, for even values of N, are symmetric about the xy plane. Everything else is the same as the simulations in Figure 4.13. As we said earlier, in the case of xy plane symmetry, we did not have to change the locations for any of the clusters for different values of N. This leads to more consistency between the simulations as N is varied.



(a) $N = 2$ (b) $N = 4$ (c) $N = 6$

(d) $N = 8$ (e) $N = 10$ (f) $N = 12$

(g) $N = 14$ (h) $N = 16$

Figure 4.14: In these figures we plot u_1 (black) , u_2 (red), u_3 (blue), and u_4 (green) at specific points in the domain. Everything is the same as in Figure 4.13 except that now the geometry for the cluster locations is that depicted in Figure 4.5. The cluster locations are symmetric about the xy plane.

In Figure 4.14 we see that changing the number of clusters can have a dramatic effect on the solutions of (4.1). Clearly the placement and size of the clusters is

important. The placement of the subcellular compartments is also important too because we have chosen larger decay rates than in Chapters 2 and 3. We saw in Chapters 2 and 3, that if an $O(\varepsilon)$ decay rate was assumed, the distance between the compartments would not have an effect on the solution to leading order. When the decay is larger this is not the case, as observed in §2.4.1. There could be many relationships between the clusters and compartments that could be further studied if some more analysis was done for the model with cell surface receptors.

We leave the discussion of the results in this chapter to the last chapter in this thesis, Chapter 5.

Chapter 5

Discussion

In this thesis we have constructed and analyzed a signal transduction model which takes into account 3D geometry, 3D diffusion, and spatial separation and localization of opposing enzymes. Using asymptotic methods we obtained steady state solutions and determined their stability. We also obtained asymptotic time dependent solutions. With the assumption that $\alpha_u^2 = O(\varepsilon)$ and $\alpha_v^2 = O(\varepsilon)$, the model in this thesis was analyzed while leaving $F(u, v)$ and $G(u, v)$ as arbitrary functions. In this scenario the concentration levels decay algebraically away from the compartments. In this case the dynamics happen on a slower time scale. It takes longer for concentrations to reach an equilibrium state which is often the final state of the system. Moreover, the solutions to the model are relatively constant in space away from the compartments. Therefore, when doing the method of matched asymptotics, all the asymptotic expansions are $O(1)$ to leading order. This is independent of the choice of functions for F and G. This is the reason the scaling, $\alpha_u = O(\sqrt{\varepsilon})$ and $\alpha_v = O(\sqrt{\varepsilon})$, was pursued in this thesis.

Another case to consider is when both $\alpha_u = O(1)$ and $\alpha_v = O(1)$. In this scenario the dynamics happen on a faster time scale. Although the analysis in this thesis can be applied to specific instances in this case, it can not be used in general while leaving the enzyme kinetic functions F and G arbitrary. This is because the scaling of the asymptotic solutions are dependent on the particular choice of functions F and G (see [94, 61]). As an example, in the case that $F = u/(v + 1)$ and $G = u^2$ with $\alpha_u = O(1)$ and $\alpha_v = O(1)$, the scaling of u is $O(1)$ and the scaling of v is $O(\varepsilon)$. If the functions F and G were changed then the scaling for u and v could very well change. This difficulty prevents us from doing the asymptotic analysis with and without delay, for arbitrary functions F and G when $\alpha_u = O(1)$ and $\alpha_v = O(1)$.

In summary, for this paper we assumed that $\alpha_u = \alpha_1\sqrt{\varepsilon}$ and $\alpha_v = \alpha_2\sqrt{\varepsilon}$ where both α_1 and α_2 are O(1) constants. This leads to long–range signalling gradients and

also dynamics which occur on a longer time scale. This time scale is of magnitude $O(1/\varepsilon)$. Here we are assuming that diffusion is sufficiently fast or the cytosolic deactivation rate is very weak. This leads to concentration gradients which are basically constant throughout the cytosol away from the compartments.

In §2.4 we showed how the full BVP model in (2.3) can be approximated by a system of DAEs. It is interesting to see how the flux boundary condition terms from (2.3) appear in the system of DAEs found in (2.47). They do not show up as the production functions $F(u_0, v_0)$ and $G(u_0, v_0)$ but rather as $c_1(u_0, v_0)$ and $c_2(u_0, v_0)$ where c_1 and c_2 are implicitly defined by $c_1 = F(u_0 + c_1, v_0)$ and $c_2 = G(u_0, v_0 + c_2)$. These relationships can be solved to turn the DAEs into ODEs. The resulting system of ODEs from (2.47), which approximates the dynamics of (2.3) in the outer region, can look very different from other ODE models modelling the same kinetics (see §2.5.4). The approximating DAE model from (2.47) comes about from the full PDE model with boundary conditions in (2.3). Spatial terms do not appear in (2.47) explicitly, but it is space, diffusion, and spatial separation of enzymes which give the DAEs their specific form as determined through asymptotic matching.

Also in §2.4 we obtained a second system of ODEs in (2.50) where the spatial terms do show up explicitly. Using the solutions from this system we obtained asymptotic expressions for the approximate time dependent solutions of (2.3) over the entire domain. These time dependent solutions, found in (2.51), are valid when the solution is evolving slowly or is close to reaching equilibrium.

There are several things to consider for future work with regards to the model in Chapter 2. One is the further effects of cell shape and geometry. In 3D, a spherical domain is the obvious choice because it simplifies the analysis. It would be interesting to investigate similar models to the one in this chapter for irregular shaped 3D geometries. If Green's functions were to be used then they would need to be found numerically. It may also be interesting to consider models where both the kinase and phosphatase enzymes are localized to different subcellular structures instead of having the phosphatase enzymes uniformly throughout the cytosol. Nonlinear PDEs, which could incorporate more general deactivation rates, would also be interesting to take a look at.

In [61], [94], and in Chapter 2, spatial clustering of compartments was not considered. It is assumed that the compartments are all an $O(1)$ distance apart. If the distance between compartments is $O(\varepsilon)$ then this will change the dynamics. The signalling proteins activated at cellular compartments would be affected by the near field behavior of other signalling proteins activated at nearby compartments. As a result the two algebraic equations in (2.47) would now become $c_1 = F(u_0+c_1, v_0+c_2)$ and $c_2 = G(u_0+c_1, v_0+c_2)$. This would allow for more complex signalling dynamics because there is now a coupled system of equations that determine the functions $c_1(u_0, v_0)$ and $c_2(u_0, v_0)$.

Another project we would like to consider is cell signalling models which use a different type of diffusion process. In the models in this thesis, the signalling molecules move around the cell using regular diffusion. Sub-diffusion has been observed in biological systems where the diffusion is often hindered due to the complex structure of the cell. There is evidence that sub-diffusion may be more accurate in modelling cell signal transduction pathways [70]. Incorporating sub-diffusion involves the use of fractional calculus. The fractional derivative is a generalization of the usual integer derivative to an arbitrary order. Using this framework the usual time derivative operator, $\frac{\partial u}{\partial t}$, would be replaced by

$$\frac{d^\alpha}{dt}u(x,t) = \frac{1}{\Gamma(1-\alpha)}\frac{d}{dt}\int_0^t \frac{u(x,s)}{(t-s)^\alpha}\,ds.$$

There is certainly no software available to solve 3D PDEs with this type of diffusion operator. Therefore numerical analysis combined with computer programming would be a large part of this work. It is unclear what analytic techniques would be appropriate at this point.

In Chapter 3 a 3D model of signal transduction with delay was analyzed. The model consisted of the model in Chapter 2 with an added constant time delay. On the boundaries of the cellular compartments, the enzyme kinetic reactions were modelled with time delayed functions. This model stands out from other signalling models because it incorporates PDEs and delay in 3D. To the best of our knowledge, there are no other mathematical models which incorporate PDEs, 3D geometry, and delay simultaneously. With regard to signal transduction models, little work has been done on models with delay. When delay is used with PDEs in mathematical models in

general, it is often in lower spatial dimensions. In our model, since the delay only appears in boundary conditions, numerical simulations of the entire 3D model with delay are more practicable.

For the model with delay, the stability of the equilibrium solutions were determined by a nonlinear eigenvalue problem with PDEs. Using matched asymptotics this eigenvalue problem was approximated by the nonlinear matrix eigenvalue problem in (3.18). By analyzing this matrix and its eigenvalues, complex signalling dynamics were found and observed such as sustained oscillations. Whether or not sustained oscillations occur depends on the value of the delay. We also saw that the delay term could change the basin of attraction for a bistable model, as in §3.5.1.

In our analysis we approximated the dynamics of the delay PDE model by the dynamics of a system of DDAEs in (3.42). Therefore, instead of doing finite element calculations in Comsol, alternative calculations using only the ODE solvers in Matlab could be done. This saves time when doing the numerical simulations since our method of steps code in Comsol takes a long time to run. The numerical simulations of the DDAEs are much quicker because no spatial configuration is required. Our implementation of the DDAE solver is still much slower than state of the art DDE code. The DDAEs are also useful for determining parameter choices for different bifurcations. For example, it is quicker to find or verify a Hopf-bifurcation using the DDAEs. The bifurcation can then be further verified by showing it also occurs in simulations of the 3D model.

The analysis in §3.6 is an interesting use of the Poincaré-Lindstedt method. Often the Poincaré-Lindstedt method is applied to ODEs. In this chapter the method is applied to PDEs in 3D with delay. After rescaling time we also had to use the method of matched asymptotics to arrive at a solvability condition for ω_2 in 3.53. The solvability condition for ω_2 in (3.53) relies heavily on the construction of the operator $\mathcal{L}$ defined in (3.48). Most of the functions in the solvability condition either are in the kernel of $\mathcal{L}$ or the kernel of its adjoint, $\mathcal{L}^*$, which is defined in (3.52).

We note that there are different ways of constructing the operator $\mathcal{L}$. Initially we differentiated the expressions for A_1 and A_2 from (3.47) so that the operator $\mathcal{L}$ would consist of four differential equations, two of them now being neutral delay ones. This increases the dimension of the kernel by two though since the vectors $(0,0,1,0)^\top$

and $(0,0,0,1)^\top$ are now both in the kernel of $\mathcal{L}^*$. Although these solutions are non trivial and periodic, they do not contribute anything useful to the corresponding solvability condition. Although this approach should work in theory, the numerical methods described in §3.6.1 were unsuccessful in finding accurate approximations to the kernel of $\mathcal{L}^*$. The resulting eigenvalue problem is ill conditioned because the kernel now has dimension three. We spent a lot of time trying this operator as opposed to the operator defined in (3.48). We thought that the operator needed to be a system of neutral DDEs. It was not until much later that we considered using the DDAE operator in (3.48).

The operator $\mathcal{L}$ defined in (3.48) came from the equations in (3.47). The next step in constructing $\mathcal{L}$ was to leave the equations for A_1 and A_2 just as they were and define the operator as a system of DDAEs. It turns out that if the expressions for A_1 and A_2 are not substituted directly into the differential equations for χ_1 and χ_2 in (3.47), then the techniques from §3.6.1 do not work. In this case the operator $\mathcal{L}$ and its discretized matrix $\overline{\boldsymbol{L}}$ have simpler forms which is a nice feature. However, solving for the kernel of $\overline{\boldsymbol{L}}$ is difficult since the eigenvalue problem is ill conditioned when this matrix is used. We are not sure why this is but we had no success in finding the kernel of $\mathcal{L}^*$ in this case. Matlab did return a warning saying that the solver could not converge because the matrix was ill conditioned. Finally, we substituted the expressions for A_1 and A_2 into the differential equations for χ_1 and χ_2 to lead to the operator $\mathcal{L}$ defined in (3.48). In this case everything worked out nicely.

For future work we would like to add the delay to other places while still leaving it in the boundary conditions. For example, we could add the delay to the decay terms in the PDEs. The matched asymptotic analysis is straight forward in this case. The stability matrix changes slightly and the DDAEs will have delay in the decay terms. The main complication of adding delay to the PDE is in doing the numerical simulations of the full system. Now history data needs to be saved in time but also at spatial points across the domain. Interpolation is then needed for a function of three spatial variables and one time variable. The addition of delay to the PDEs will lead to richer dynamics as well as more potential biological applications.

We are also interested in studying the model in this chapter with multiple delays. For example, we could use four different delays in the boundary conditions. The

enzyme kinetic functions would then be $F(u(t-s_1), v(t-s_2))$ and $G(u(t-s_3), v(t-s_4))$. With multiple delays the analysis in this chapter can still be carried out but the numerical simulations also become more challenging. In the case of multiple delays it would be interesting to find more complex dynamics such as double Hopf-bifurcations.

In Chapter 4 we considered an extension of the model from Chapter 2 by adding in cell surface receptors. In this model the signal transduction pathway can be initiated at the cell surface. The most difficult part of this chapter was actually getting the Comsol simulations to work because it did not allow for the receptors to intersect specific lines on the sphere. Apparently the geometry used in this chapter is too complicated for the default Comsol CAD kernel.

The work in Chapter 4 was more preliminary and no actual analysis was done. From some of our numerical simulations we can see the effects of changing the number of receptors on the model. From §4.3 we observed that increasing the number of receptor clusters led to a more uniform signal across the cellular domain. This effect was more evident when the decay rate, due to the phosphatase enzymes, was stronger. In §4.4, we considered a model that had oscillations. The oscillations either decayed or continued indefinitely, depending on the number of clusters. It is apparent that the size, location, and number of clusters can have a dramatic effect on the dynamics of signal transduction pathways.

The next step for the models presented in Chapter 4 would be to do some actual analysis like that done in Chapter 2. If we considered a model with N patches where the radius of each patch was sufficiently small then we could employ some asymptotic methods. The method of matched asymptotics could be used and we would require different Green's functions such as the ones used in [21, 81]. We could most likely find the steady state solutions, do the stability analysis, and also do the same type of reduction from the PDE models to some type of ODE models. It would also be interesting to consider the analysis if the patches were larger.

More complicated models where the clusters are not assumed to be in a stationary state would also be interesting. Models where the receptors are allowed to move around dynamically in time and cluster, combined along with the 3D geometry and internal compartments, would be very nice. Numerical simulations of this type

though would require some sort of complicated moving mesh mechanism in 3D. This sounds like a very interesting but difficult problem both analytically and especially numerically.

Bibliography

[1] *COMSOL Multiphysics (v3.5a)*, COMSOL AB, Stockholm, 2007.

[2] *Maple v(13)*, Maplesoft, Waterloo, 2009.

[3] *COMSOL Multiphysics (v4.4)*, COMSOL AB, Stockholm, 2013.

[4] D. Angeli, J. Ferrell, and E. Sontag. Detection of multistability, bifurcations, and hysteresis in a large class of biological positive-feedback systems. *Proc. Natl. Acad. Sci.*, 101:1822–1827, 2004.

[5] C. Baker and C. Paul. Discontinuous solutions of neutral delay differential equations. *Appl. Numer. Math.*, 56:284–304, 2006.

[6] G. Barton. *Elements of Green's Functions and Propagation*. Oxford Science Publications. 1989.

[7] M. Behar, N. Hao, H Dohlman, and T. Elston. Mathematical and computational analysis of adaptation via feedback inhibition in signal transduction pathways. *Biophys. J*, 93:806–821, 2007.

[8] A. Bellen and M. Zennaro. *Numerical Methods for Delay Differential Equations*. Oxford University Press, 2013.

[9] A. Berezhkovskii, M. Coppey, and S. Shvartsman. Signaling gradients in cascades of two-state reaction-diffusion systems. *Proc. Natl. Acad. Sci.*, 106:1087–1092, 2009.

[10] B. Binder and R. Heinrich. Interrelations between dynamical properties and structural characteristics of signal transduction networks. *Genome. Inform. S.*, 15:13–23, 2004.

[11] N. Blüthgen and H. Herzel. How robust are switches in intracellular signaling cascades? *J. Theor. Biol.*, 225:293–300, 2003.

[12] D. Bray and T. Duke. Conformational spread: The propagation of allosteric states in large multiprotein complexes. *Annu. Rev. Bioph. Biom.*, 33:53–73, 2004.

[13] G. Brown and B. Kholodenko. Spatial gradients of cellular phospho-proteins. *FEBS Lett.*, 457:452 – 454, 1999.

[14] R. Bürger, R. Ruiz-Baier, and C. Tian. Stability analysis and finite volume element discretization for delay-driven spatial patterns in a predator-prey model. *J. Math. Anal. Appl. submitted*, 2013.

[15] J. Butcher. *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*. Wiley-Interscience, 1987.

[16] B. Care and H. Soula. Impact of receptor clustering on ligand binding. *BMC Syst. Biol.*, 5:48, 2011.

[17] B. Caré and H. Soula. Receptor clustering affects signal transduction at the membrane level in the reaction-limited regime. *Phys. Rev. E*, 87:012720, 2013.

[18] B. Casar and et al. Ras subcellular localization defines extracellular signal-regulated kinase 1 and 2 substrate specificity through distinct utilization of scaffold proteins. *Mol. Cell Biol.*, 29:1338–1353, 2009.

[19] M. Cebecauer, M. Spitaler, A. Serg, and A. Magee. Signalling complexes and clusters: functional advantages and methodological hurdles. *J. Cell Sci.*, 123:309–320, 2010.

[20] M. Chaves, E. Sontag, and R. Dinerstein. Optimal length and signal amplification in weakly activated signal transduction cascades. *J. Phys. Chem. B*, 108:15311–15320, 2004.

[21] A. Cheviakov, M. Ward, and R. Straube. An asymptotic analysis of the mean first passage time for narrow escape problems: Part ii: The sphere. *Multiscale Model. Sim.*, 8:836–870, 2010.

[22] S. Corwin, D. Sarafyan, and S. Thompson. Dklag6: a code based on continuously imbedded sixth-order runge-kutta methods for the solution of state-dependent functional differential equations. *Appl. Numer. Math.*, 24:319–330, 1997.

[23] E. de Jager and J. Furu. *The theory of singular perturbations*. Elsevier, 1996.

[24] T. Duke and D. Bray. Heightened sensitivity of a lattice of membrane receptors. *P. Natl. Acad. Sc.*, 96:10104–10108, 1999.

[25] T. Duke and I. Graham. Equilibrium mechanisms of receptor clustering. *Prog Biophys. Mol. Bio.*, 100:18 – 24, 2009.

[26] W. Enright and H. Hayashi. A delay differential equation solver based on a continuous runge–kutta method with defect control. *Numer. Algorithms*, 16:349–364, 1997.

[27] J. Ferrell and E. Machleder. The biochemical basis of an all-or-none cell fate switch in xenopus oocytes. *Science*, 280:895–898, 1998.

[28] J. Ferrell and W. Xiong. Bistability in cell signaling: How to make continuous processes discontinuous, and reversible processes irreversible. *Chaos: An Inter. J. of Non. Sci.*, 11:227–236, 2001.

[29] B. Fuller, M. Lampson, E. Foley, S. Rosasco-Nitcher, K. Le, P. Tobelmann, D. Brautigan, P. Stukenberg, and T. Kapoor. Midzone activation of aurora b in anaphase produces an intracellular phosphorylation gradient. *Nature*, 453:1132–1136, 2008.

[30] P. Garca-Pearrubia and J. Glvez. Mathematical modelling and computational study of two-dimensional and three-dimensional dynamics of receptor–ligand interactions in signalling response mechanisms. *J. Math. Biol.*, 69:1–30, 2013.

[31] C. Gear and O. Osterby. Solving ordinary differential equations with discontinuities. *ACM Trans. on Mat. Soft.*, 10:23–44, 1984.

[32] A. Goldbeter and D. Koshland. An amplified sensitivity arising from covalent modification in biological systems. *P. Natl. Acad. Sc.*, 78:6840–6844, 1981.

[33] A. Goldbeter and D. Koshland. Sensitivity amplification in biochemical systems. *Q. Rev. Biophys.*, 15:555–591, 1982.

[34] A. Goldbeter and D. Koshland. Ultrasensitivity in biochemical systems controlled by covalent modification. interplay between zero-order and multistep effects. *J. of Biol. Chem.*, 259:14441–14447, 1984.

[35] M. Golubitsky, D. Schaeffer, and I. Stewart. *Singularities and groups in bifurcation theory*. Springer New York, 1988.

[36] K Gordon, I Van Leeuwen, S. Lain, and M. Chaplain. Spatio-temporal modelling of the p53–mdm2 oscillatory system. *Math. Model. of Nat. Phen.*, 4:97–116, 2009.

[37] C. Govern and A. Chakraborty. Signaling cascades modulate the speed of signal propagation through space. *PloS one*, 4:e4639, 2009.

[38] W. Gu and P. Wang. A crank-nicolson difference scheme for solving a type of variable coefficient delay partial differential equations. *J. Appl. Math.*, 2014, 2014.

[39] R. Heinrich, B. Neel, and T. Rapoport. Mathematical models of protein kinase signal transduction. *Molecular Cell*, 9:957–970, 2002.

[40] R. Hinch and S. Schnell. Mechanism equivalence in enzyme–substrate reactions: Distributed differential delay in enzyme kinetics. *J. Math Chem*, 35:253–264, 2004.

[41] D. Holcman and Z. Schuss. Escape through a small opening: Receptor trafficking in a synaptic membrane. *J. Stat. Phys*, 117:975–1014, 2004.

[42] D. Holcman and Z. Schuss. Diffusion escape through a cluster of small absorbing windows. *J. Phys A-Math Theor.*, 41:155001, 2008.

[43] C. Huang and J. Ferrell. Ultrasensitivity in the mitogen-activated protein kinase cascade. *Proc. Natl. Acad. Sci.*, 93:10078–10083, 1996.

[44] C. Huang and S. Vandewalle. Unconditionally stable difference methods for delay partial differential equations. *Numer. Math.*, 122:579–601, 2012.

[45] P. Kalab, K. Weis, and R. Heald. Visualization of a ran-gtp gradient in interphase and mitotic xenopus egg extracts. *Science*, 295:2452–2456, 2002.

[46] U. Kent, S. Mao, C. Wofsy, B. Goldstein, S. Ross, and H. Metzger. Dynamics of signal transduction after aggregation of cell-surface receptors: Studies on the type i receptor for IgE. *P. Natl. Acad. Sc.*, 91:3087–3091, 1994.

[47] B. Kholodenko. Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades. *Eur. J. Biochem.*, 267:1583–1588, 2000.

[48] B. Kholodenko. Map kinase cascade signaling and endocytic trafficking: a marriage of convenience? *Trends in Cell Biol.*, 12:173–177, 2002.

[49] B. Kholodenko. Cell-signalling dynamics in time and space. *Nat. Rev. Mol. Cell. Biol.*, 11:165–176, 2006.

[50] B. Kholodenko. Spatially distributed cell signalling. *FEBS Lett.*, 583:4006–4012, 2009.

[51] B. Kholodenko, G. Brown, and Hoek J. Diffusion control of protein phosphorylation in signal transduction pathways. *Biochem. J*, 350:901–907, 2000.

[52] B. Kholodenko, O. Demin, G. Moehren, and J. Hoek. Quantification of short term signaling by the epidermal growth factor receptor. *J. of Biol. Chem.*, 274:30169–30181, 1999.

[53] B. Kholodenko, J. Hancock, and W. Kolch. Signalling ballet in space and time. *Nat. Rev. Mol. Cell. Biol.*, 11:414–426, 2010.

[54] B. Kholodenko and W. Kolch. Giving space to cell signaling. *Cell*, 133:566–567, 2008.

[55] E. Klipp and W. Liebermeister. Mathematical modeling of intracellular signaling pathways. *BMC Neurosci.*, 7:S10, 2006.

[56] C. Koay. Distributing points uniformly on the unit sphere under a mirror reflection symmetry constraint. *J. of Comput. Sci.*, 5:696 – 700, 2014.

[57] H. Kobayashi, R. Azuma, and A. Konagaya. Clustering of membrane proteins in the pre-stimulation stage is required for signal transduction: A computer analysis. *Signal Transduction*, 7:329–339, 2007.

[58] W. Kolch. Coordinating erk/mapk signalling through scaffolds and inhibitors. *Nat. Rev. Mol. Cell. Biol.*, 6:827–837, 2005.

[59] D. Koshland, A. Goldbeter, and J. Stock. Amplification and adaptation in regulatory and sensory systems. *Science*, 217:220–225, 1982.

[60] M. Laurent and N. Kellershohn. Multistability: a major means of differentiation and evolution in biological systems. *Trends Biocem. Sci.*, 24:418–422, 1999.

[61] C. Levy and D. Iron. Model of cell signal transduction in a three-dimensional domain. *J. Math. Biol.*, 63:831–854, 2011.

[62] C. Levy and D. Iron. Dynamics and stability of a three-dimensional model of cell signal transduction. *J. Math. Biol.*, 67:1691–1728, 2013.

[63] C. Levy and D. Iron. Dynamics and stability of a three-dimensional model of cell signal transduction with delay. *Nonlinearity. submitted*, 2014.

[64] T. Lewis, P. Shapiro, and N. Ahn. Signal transduction through map kinase cascades. *Adv. Cancer Res.*, 74:49–139, 1998.

[65] D. Li and C. Zhang. On the long time simulation of reaction-diffusion equations with delay. *Sci. World J.*, 2014, 2014.

[66] D. Li, C. Zhang, and H. Qin. Ldg method for reaction–diffusion dynamical systems with time delay. *Appl. Math. Comput.*, 217:9173–9181, 2011.

[67] D. Li, C. Zhang, and J. Wen. A note on compact finite difference method for reaction–diffusion equations with delay. *Appl. Math. Model.*, 2014.

[68] Y. Li and J. Srividhya. Goldbeter–Koshland model for open signaling cascades: a mathematical study. *J. Math. Biol.*, 61:781–803, 2010.

[69] Y. Li and J. Srividhya. Goldbeter-Koshland model for open signaling cascades: a mathematical study. *J. Math. Biol.*, 61:781–803, 2010.

[70] Weissm M., M. Elsner, F. Kartberg, and T. Nilsson. Anomalous subdiffusion is a measure for cytoplasmic crowding in living cells. *Biophys. J.*, 87:3518–3524, 2004.

[71] P. Mandel and T. Erneux. The slow passage through a steady bifurcation: Delay and memory effects. *J. Statist. Phys.*, 48:1059–1070, 1987.

[72] N. I Markevich, J. Hoek, and B. Kholodenko. Signaling switches and bistability arising from multisite phosphorylation in protein kinase cascades. *J. Cell Biol.*, 164:353–359, 2004.

[73] Nick. Markevich, M. Tsyganov, J. Hoek, and Boris. Kholodenko. Long-range signaling by phosphoprotein waves arising from bistability in protein kinase cascades. *Mol. Syst. Biol.*, 2:61, 2006.

[74] J. Meyers, J. Craig, and D. Odde. Potential for control of signaling pathways via cell size and shape. *Curr. Biol.*, 16(17):1685–1693, 2006.

[75] I. Morrison, C. M Anderson, G. Georgiou, G. Stevenson, and R. Cherry. Analysis of receptor clustering on cell surfaces by imaging fluorescent particles. *Biophys. J*, 67:1280 – 1290, 1994.

[76] J. Moseley, A. Mayeux, A. Paoletti, and P. Nurse. A spatial gradient coordinates cell size and mitotic entry in fission yeast. *Nature*, 459:857–860, 2009.

[77] T. Naka, M. Hatakeyama, N. Sakamoto, and A. Konagaya. Compensation effect of the MAPK cascade on formation of phospho-protein gradients. *Biosystems*, 83:167 – 177, 2006.

[78] K. Neves. Automatic integration of functional differential equations: An approach. *ACM Trans. on Mat. Soft.*, 1:357–368, 1975.

[79] S. Neves and R. Iyengar. Models of spatially restricted biochemical reaction systems. *J. Biol. Chem.*, 284:5445–5449, 2009.

[80] F. Ortega, J. Garcs, F. Mas, B. Kholodenko, and M. Cascante. Bistability from double phosphorylation in signal transduction. *FEBS J.*, 273:3915–3926, 2006.

[81] S. Pillay, M. Ward, A. Peirce, and T. Kolokolnikov. An asymptotic analysis of the mean first passage time for narrow escape problems: Part i: Two-dimensional domains. *Multiscale Model. Sim.*, 8:803–835, 2010.

[82] L. Qiao, R. Nachbar, I. Kevrekidis, and S. Shvartsman. Bistability and oscillations in the huang-ferrell model of mapk signaling. *PLoS Comput. Biol.*, 3:e184, 2007.

[83] Z. Qu and T. Vondriska. The effects of cascade length, kinetics and feedback loops on biological signal transduction dynamics in a simplified cascade model. *Phys. Biol.*, 6:016007, 2009.

[84] J. Saez-Rodriguez, A. Kremling, H. Conzelmann, K. Bettenbrock, and E. Gilles. Modular analysis of signal transduction networks. *Control Syst., IEEE*, 24:35–52, 2004.

[85] C. Salazar and R. Höfer. Kinetic models of phosphorylation cycles: A systematic approach using the rapid-equilibrium approximation for protein–protein interactions. *Biosyst.*, 83:195–206, 2006.

[86] R. Sear and M. Howard. Modeling dual pathways for the metazoan spindle assembly checkpoint. *P. Natl. Acad. Sc.*, 103:16758–16763, 2006.

[87] L. Shampine. Solving odes and ddes with residual control. *Appl. Numer. Math.*, 52:113–127, 2005.

[88] L. Shampine. Dissipative approximations to neutral ddes. *Appl. Math. Comput.*, 203:641–648, 2008.

[89] L. Shampine, I. Gladwell, and S. Thompson. *Solving ODEs with MATLAB*. Cambridge University Press, 2003.

[90] L. Shampine and S. Thompson. Solving ddes in matlab. *Appl. Numer. Math.*, 37:441–458, 2001.

[91] A. Shaw and E. Filbert. Scaffold proteins and immune-cell signalling. *Nat. Rev. Immunol.*, 9:47–56, 2009.

[92] J. Srividhya, M.S. Gopinathan, and S. Schnell. The effects of time delays in a phosphorylation–dephosphorylation pathway. *Biophys. Chem.*, 125:286 – 297, 2007.

[93] J. Stelling and B. Kholodenko. Signaling cascades as cellular devices for spatial computations. *J. Math Biol.*, 58:35–55, 2009.

[94] R. Straube and M. Ward. An asymptotic analysis of intracellular signaling gradients arising from multiple small compartments. *SIAM J. Appl. Math.*, 70:248–269, 2008.

[95] M. Sturrock, A. Terry, D. Xirodimas, A. Thompson, and M. Chaplain. Spatio-temporal modelling of the hes1 and p53-mdm2 intracellular signalling pathways. *J. Theor. Biol.*, 273:15–31, 2011.

[96] B. Sulzer and A. Perelson. Equilibrium binding of multivalent ligands to cells: Effects of cell and receptor density. *Math. Bioscience.*, 135:147 – 185, 1996.

[97] A. Terry and M. Chaplain. Spatio-temporal modelling of the nf-κb intracellular signalling pathway: the roles of diffusion, active transport, and cell geometry. *J. Theor. Biol.*, 290:7–26, 2011.

[98] S. Thompson and L. Shampine. A friendly fortran dde solver. *Appl. Numer. Math.*, 56:503–516, 2006.

[99] J Thomson. Xxiv. on the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure. *Philos. Mag. Series 6*, 7:237–265, 1904.

[100] J. Tyson, K. Chen, and B. Novak. Sniffers, buzzers, toggles and blinkers: Dynamics of regulatory and signaling pathways in the cell. *Curr. Opin. Cell Biol.*, 15:221–231, 2003.

[101] S. Van Albada and P. ten Wolde. Enzyme localization can drastically affect signal amplification in signal transduction pathways. *PLoS Comput. Biol.*, 3:e195, 2007.

[102] P. Van der Houwen, B. Sommeijer, and C. Baker. On the stability of predictor-corrector methods for parabolic equations with delay. *IMA J. of Numer. Anal.*, 6:1–23, 1986.

[103] M. Van Dyke. *Perturbation Methods in Fluid Mechanics*. Applied Mathematics and Mechanics, Vol. 8. 1964.

[104] A. Ventura, J. Sepulchre, and S. Merajver. A hidden feedback in signaling cascades is revealed. *PLoS Comput. Biol.*, 4:e1000041, 2008.

[105] S. Wanant and M. Quon. Insulin receptor binding kinetics: Modeling and simulation studies. *J. of Theor. Biol.*, 205:355 – 364, 2000.

[106] C. Widmann, S. Gibson, M. Jarpe, and G. Johnson. Mitogen-activated protein kinase: conservation of a three-kinase module from yeast to human. *Phys. Rev.*, 79:143–180, 1999.

[107] O. Wolkenhauer, S. Sreenath, P. Wellstead, M. Ullah, and K. Cho. A systems- and signal-oriented approach to intracellular dynamics. *Biochem. Soc. T.*, 33:507–515, 2005.

[108] J. Xing and J. Chen. The Goldbeter–Koshland switch in the first-order region and its response to dynamic disorder. *PloS one*, 3:e2140, 2008.

[109] J. Xing and J. Chen. The Goldbeter-Koshland switch in the first-order region and its response to dynamic disorder. *PLoS One*, page e2140, 2008.

[110] W. Xiong and J. Ferrell. A positive-feedback-based bistable ?memory module?that governs a cell fate decision. *Nature*, 426:460–465, 2003.

[111] I. Yudushkin, A. Schleifenbaum, A. Kinkhabwala, B. Neel, C. Schultz, and P. Bastiaens. Live-cell imaging of enzyme-substrate interaction reveals spatial regulation of ptp1b. *Science*, 315:115–119, 2007.

[112] Q. Zhang and C. Zhang. A new linearized compact multisplitting scheme for the nonlinear convection–reaction–diffusion equations with delay. *Commun. Nonlinear. Sci.*, 18:3278–3288, 2013.

[113] Q. Zhang, C. Zhang, and D. Deng. Compact alternating direction implicit method to solve two-dimensional nonlinear delay hyperbolic differential equations. *Int. J. Comput. Math.*, 91:1–19, 2013.

Appendix A

Comsol and Matlab Code

A.1 Standard DDE Solvers

Here we discuss some important ideas which are involved in creating efficient numerical solvers for DDEs. We also mention several specific solvers which are commonly used to solve standard DDEs numerically. There are different types of DDEs and each have different properties with regards to discontinuity propagation and smoothing. A very large class of DDEs can be written in the form

$$y'(t) = f(t, y(t), y(t - T_1), y(t - T_2), \ldots, y(t - T_k)). \tag{A.1}$$

For DDEs of the form (A.1) with multiple delays, a discontinuity of order k at time $t = t^*$ is propagated to a discontinuity of at least order $k + 1$ at times

$$t^* + T_1, t^* + T_2, \ldots, t^* + T_k.$$

The most common case in modelling literature are when the delay terms T_j (also called lag functions) are constants [89, 90]. However there is interest in systems of DDEs where the lag functions are time dependent and or state dependent. This means that the lag functions would have the form $T_j(t, y(t))$. There are also DDEs which have delayed arguments appearing in the derivative terms. These are called neutral DDEs. They can have very different behavior and pose further theoretical and numerical difficulties. For example, the solutions to neutral DDEs may not get smoother as the integration proceeds [5].

There are some main features that any robust numerical DDE solver should have. One is the ability to deal with the derivative discontinuities. When discontinuities are present much care must be taken in writing efficient and accurate numerical algorithms. Discontinuities can affect the local truncation error analysis. The local truncation error, which depends on higher derivatives within a Taylor expansion, will be bounded provided that the solution is smooth. If there are discontinuities, the

higher derivatives are not bounded, and hence the local truncation error analysis is no longer valid. This has a direct impact on the convergence of a method since the order of convergence can be lowered in the presence of discontinuities. This is known as order reduction.

One method for dealing with discontinuities is to ignore them and assume that the local error estimator along with the adaptive time stepping algorithm will choose a small enough step size near the discontinuity. For larger discontinuities a smaller step size is needed. A discretization in time which ignores discontinuities can be inefficient. The adaptive time stepping algorithm will continue to force a very small time step until it can eventually step over the discontinuity with a small enough time step that satisfies the error tolerance [31].

Another method is to locate discontinuities in time and include them as step points. It is trivial to locate discontinuities in the case where the lag functions are constants. In this case all the points of discontinuity are known in advance and can be included as step points through the adaptive time stepping algorithm. It is clearly much harder when the lag functions are state dependent because $y(t)$ is unknown and therefore must be approximated. If there is a discontinuity at $t = t^*$ in (A.1) then this discontinuity will in general propagate if the delayed arguments satisfy $t - T_j(t, y(t)) = t^*$. The propagated discontinuities can be located in time by solving equations of the form

$$t - T_j(t, y(t)) - t^* = 0, \tag{A.2}$$

for t. The zeros of (A.2) can be approximated numerically.

Since the solution of a system of DDEs becomes smoother as the integration advances, there will be a point in time when the tracking of discontinuities can stop. This is when the discontinuities have reached a high enough order as not to interfere with the local truncation error and hence the order of convergence of the method. Only discontinuities in lower order derivatives need to be kept track of because they affect the local truncation error analysis.

Another main feature of any robust DDE code is the ability to evaluate the delayed arguments efficiently and accurately. DDE codes should have continuous output due to the delayed arguments which access previous solution values. This is done through some form of interpolation. It is important that the interpolation

method preserves the order of accuracy of the integration method.

The first code written for DDEs was called **DMRODE** [78] and was written in Fortran 77. The solver consists of a basic ODE solver routine and relied on the automatic step change algorithm to step through discontinuities. This is a very inefficient process. All the computed solution values at step points are saved. Hermite interpolation is used to calculate the delayed arguments. The code does not allow for systems with multiple delays to be directly entered.

A code based on DMRODE was **DKLAG5** which was eventually followed by **DKLAG6** [22]. **DKLAG6** was written in Fortran 77. **DKLAG6** can handle DDEs with state dependent delays as well as neutral differential equations. It can also solve DDEs with vanishing or near vanishing lag functions. The code uses a continuously embedded sixth order Rung-Kutta (RK) method. The code **DKLAG6** locates derivative discontinuities in time by using a root finder along with the polynomial approximants. All points of discontinuity are included as step points.

Although **DKLAG6** can solve a large class of problems, many users find it hard to use because it is written in Fortran. The code **DDE_SOLVER** was written to address the complexity of using **DKLAG6**. It was written in Fortran 90/95 and is essentially the same as **DKLAG6** except there is a more friendly user interface. The improvements of **DDE_SOLVER** over **DKLAG6** can be found in [98].

The code **DDVERK** [26] is a DDE code which was specifically designed to solve neutral DDEs. It uses continuous RK methods and uses a discontinuity detection algorithm based on defect control to locate discontinuities. The defect of the continuous approximation is a measure of the amount by which the approximate solution fails to satisfy the differential equation. By monitoring the defect at every step, it is possible to locate points of discontinuity and include them as step points.

Another code to solve neutral DDEs is **ddeNsd** [88]. It is written in Matlab and is therefore relatively easy to use because of its simple interface. The code approximates neutral DDEs with standard (non neutral) DDEs, which have smoother solutions and therefore are easier to solve. The code is based on the two Matlab codes **dde23** (see below) and **ddesd** [87] which solve DDEs with constant delays and DDEs with state dependent delays respectively. This code **ddeNsd** can only solve neutral DDEs with modest accuracy. We have experimented with **ddeNsd** and discuss this in §3.4.3.

The final DDE code we mention here is **dde23** [90]. This is a very popular code because it is written in Matlab and is easy to use. The goal of the code is to make it straight forward to solve DDE systems of the form

$$y'(t) = f(t, y(t), y(t - T_1), t(t - T_2), \ldots, y(t - T_k)), \tag{A.3}$$

where the lag functions are positive constants. The authors of **dde23**, L.F. Shampine and S. Thompson, are well aware of other types of lag functions and in fact have developed some of the more general codes mentioned above. They note that constant delays are the most frequently encountered lag functions in modelling. Restricting the software to just DDEs of the form (A.3) allows for a much simpler user interface as well as a more robust software.

A.2 Solving the PDE Model in (2.3) Without Delay

This code can solve problems of the form (2.3). The code here is for a signalling pathway with three variables and three compartments. Therefore it's an extension of (2.3) to a system of 3 PDEs. The code here was used to solve the Hopf bifurcation example in §2.5.3.

```
% COMSOL Multiphysics Model M-file
% Generated by COMSOL 3.5a (COMSOL 3.5.0.608, $Date: 2009/05/11 07:38:49 $)
flclear fem
% COMSOL version
clear vrsn
vrsn.name = 'COMSOL 3.5';
vrsn.ext = 'a';
vrsn.major = 0;
vrsn.build = 608;
vrsn.rcs = '$Name: v35ap $';
vrsn.date = '$Date: 2009/05/11 07:38:49 $';
fem.version = vrsn;
%%%%
fu1='(8*(1-u1))/(8+(.2+50*u3)^3)';
fu2='u1^3*(1-u2)/(0.8e-2+u1^3)';
fu3='u2^3*(1-u3)/(0.8e-2+u2^3)';
u1int=0;
u2int=0;
u3int=0;
epsilon=0.01;
x1=0.5;
x2=0;
x3=-0.5;
% Constants
fem.const = {'R','1', 'k1','1/2', 'k2','1', k3','1/4',  'Du1','1/3',  'Du2','1/3',  'Du3','1/3', ...
  'epsilon',num2str(epsilon), ...
  'u1int',num2str(u1int), ...
  'u2int',num2str(u2int), ...
```

```
  'u3int',num2str(u3int), ...
   'ku1','k1*epsilon', ... %Do not edit below this
  'ku2','k2*epsilon', ...
    'ku3','k3*epsilon', ...
  'tauu1','R^2/Du1', ...
  'tauu2','R^2/Du2', ...
    'tauu3','R^2/Du3', ...
    'alphau1','sqrt(ku1/Du1)*R', ...
  'alphau2','sqrt(ku2/Du2)*R', ...
  'alphau3','sqrt(ku3/Du3)*R', ...
  'alphau4','sqrt(ku4/Du4)*R'};
% Geometry
g1=sphere3('1','pos',{'0','0','0'},'axis',{'0','0','1'},'rot','0');
g2=sphere3(num2str(epsilon),'pos',{num2str(x3),'0','0'}, 'axis',{'0','0','1'},'rot','0');
g3=sphere3(num2str(epsilon),'pos',{num2str(x2),'0','0'},'axis',{'0','0','1'},'rot','0');
g4=sphere3(num2str(epsilon),'pos',{num2str(x1),'0','0'}, 'axis',{'0','0','1'},'rot','0');
g5=geomcomp({g1,g2,g3,g4},'ns',{'g1','g2','g3','g4'},'sf', 'g1-g2-g3-g4','face','none','edge','all');
% Geometry
% Analyzed geometry
clear s
s.objs={g5};
s.name={'CO1'};
s.tags={'g5'};
fem.draw=struct('s',s);
fem.geom=geomcsg(fem);
% Initialize mesh
fem.mesh=meshinit(fem, 'hauto',5);
% (Default values are not included)
% Application mode 1
clear appl
appl.mode.class = 'FlPDEC';
appl.dim = {'u1','u2','u3','u1_t','u2_t','u3_t'};
appl.gporder = 4;
appl.cporder = 2;
appl.sshape = 2;
appl.assignsuffix = '_c';
clear bnd
bnd.g = {0,{['1/epsilon*' fu1];0;0},{0;['1/epsilon*' fu2];0},{0; 0;['1/epsilon*' fu3]}};
bnd.name = {'outer','u1','u2','u3'};
bnd.type = 'neu';
bnd.ind = [1,1,1,1,4,4,4,4,4,4,4,4,3,3,3,3,1,1,3,3,1,3,3,1,2,2,2,2,2,2, 2,2];
appl.bnd = bnd;
clear equ
equ.f = 0;
equ.da = {{'tauu1';'tauu2';'tauu3'}};
equ.c = {{'1';'1';'1'}};
equ.a = {{'alphau1^2';'alphau2^2';'alphau3^2'}};
equ.init = {{'u1int';'u2int';'u3int';0;0;0}};
equ.ind = [1];
appl.equ = equ;
fem.appl{1} = appl;
fem.frame = {'ref'};
fem.border = 1;
clear units;
units.basesystem = 'SI';
fem.units = units;
% ODE Settings
clear ode
clear units;
units.basesystem = 'SI';
ode.units = units;
fem.ode=ode;
% Multiphysics
fem=multiphysics(fem);
% Extend mesh
fem.xmesh=meshextend(fem);
```

```
% Solve problem
fem.sol=femtime(fem, ...
                'solcomp',{'u2','u1','u3'}, ...
                'outcomp',{'u2','u1','u3'}, ...
                'blocksize','auto', ...
                'tlist',[colon(0,1,15000)], ...
                 'rtol',1e-6, ...
                'tout','tlist', ...
                'atol',{'1e-6'}, ...
                  'linsolver','gmres', ...
                'prepar',{'droptol',1E-5});
%Post Processing
 t=fem.sol.tlist;
u=postinterp(fem,'u1',[-1;0;0],'solnum',1:length(t));
uin=postinterp(fem,'u1',[.5+epsilon;0;0],'solnum',1:length(t));
v=postinterp(fem,'u2',[-1;0;0],'solnum',1:length(t));
vin=postinterp(fem,'u2',[epsilon;0;0],'solnum',1:length(t));
w=postinterp(fem,'u3',[1;0;0],'solnum',1:length(t));
win=postinterp(fem,'u3',[-.5-epsilon;0;0], 'solnum',1:length(t));
save comsolsim u v w uin vin win  t
```

A.3 Comsol Method of Steps Code

This code solves the PDE model with delay found in (3.2). The method of steps is used along with Comsol as explained in §3.4.1. The code here was used to solve the Hopf bifurcation example in §(3.5.2).

```
% COMSOL Multiphysics Model M-file
% Generated by COMSOL 3.5a (COMSOL 3.5.0.608, $Date: 2009/05/11 07:38:49 $)
% %load coordvectors01.mat;
flclear fem;
flclear fem0;
% COMSOL version
clear vrsn
vrsn.name = 'COMSOL 3.5';
vrsn.ext = 'a';
vrsn.major = 0;
vrsn.build = 608;
vrsn.rcs = '$Name: v35ap $';
vrsn.date = '$Date: 2009/05/11 07:38:49 $';
fem.version = vrsn;
%This code is set up for having x1 and x2 located at (0.5,0,0) and (-.5,0,0)
%These numbers can be changed by editing certain numbers through out code.
epsilon=0.1;
delay=2/epsilon;
k1=1;
k2=1;
Du=1/3;
Dv=1/3;
uint=.5;
vint=0.8;
rtol=1e-4; %rel tol
atol=1e-4; %abs tol
droptol=1e-2; %drop tol
initialstep=1e-6; %initial step (probably ignored because of tolerance)
N=29;  %Number of delay intervals to solve on
% Constants
fem.const = {'R','1', ...
```

```
  'k1',num2str(k1), ...
  'k2',num2str(k2), ...
  'Du',num2str(Du), ...
  'Dv',num2str(Dv), ...
  'epsilon',num2str(epsilon), ...
  'uint',num2str(uint), ... %  initial conditions
  'vint',num2str(vint), ...
  'ku','k1*epsilon', ...
  'kv','k2*epsilon', ...
  'tauu','R^2/Du', ...
  'tauv','R^2/Dv', ...
  'alphau','sqrt(ku/Du)*R', ...
  'alphav','sqrt(kv/Dv)*R'};
%%%%%
Fuv='4*uint/((1/2+uint)*(1+vint^2))';%These are used outside the loop when solving on first
% delay interval and using constant history data.
Guv='2*vint*uint/(1/4+vint)';
Fuvloop='4*Fin(t)/((1/2+Fin(t))*(1+G(t)^2))';  %These are the functions which show up in Loop
Guvloop='2*Gin(t)*F(t)/(1/4+Gin(t))';
% Geometry
g1=sphere3('1','pos',{'0','0','0'},'axis',{'0','0','1'},'rot','0','const',fem.const);
g2=sphere3('epsilon','pos',{'-0.5','0','0'},'axis',{'0','0','1'},'rot','0','const',fem.const);
g3=sphere3('epsilon','pos',{'0.5','0','0'},'axis',{'0','0','1'},'rot','0','const',fem.const);
g4=geomcomp({g1,g2,g3},'ns',{'g1','g2','g3'},'sf','g1-g2-g3','face','none','edge','all');
% Analyzed geometry
clear s
s.objs={g4};
s.name={'CO1'};
s.tags={'g4'};
fem.draw=struct('s',s);
fem.geom=geomcsg(fem);
% Initialize mesh
fem.mesh=meshinit(fem, ...
                  'hauto',5);

% (Default values are not included)
% Application mode 1
clear appl
appl.mode.class = 'FlPDEC';
appl.dim = {'u','v','u_t','v_t'};
appl.gporder = 4;
appl.cporder = 2;
appl.sshape = 2;
appl.assignsuffix = '_c';
clear bnd
bnd.g = {0,{['1/epsilon*' Fuv];0},{0;['1/epsilon*' Guv]}};
bnd.name = {'outer','uin','vin'};
bnd.type = 'neu';
bnd.ind = [1,1,1,1,3,3,3,3,3,3,3,1,1,1,1,2,2,2,2,2,2,2,2];
appl.bnd = bnd;
clear equ
equ.f = 0;
equ.da = {{'tauu';'tauv'}};
equ.c = {{'1';'1'}};
equ.a = {{'alphau^2';'alphav^2'}};
equ.init = {{'uint';'vint';0;0}};
equ.ind = [1];
appl.equ = equ;
fem.appl{1} = appl;
fem.frame = {'ref'};
fem.border = 1;
clear units;
units.basesystem = 'SI';
fem.units = units;
% ODE Settings
clear ode
```

```
clear units;
units.basesystem = 'SI';
ode.units = units;
fem.ode=ode;
% Multiphysics
fem=multiphysics(fem);
% Extend mesh
fem.xmesh=meshextend(fem);
% Solve problem
fem.sol=femtime(fem, ...
                'solcomp',{'v','u'}, ...
                'outcomp',{'v','u'}, ...
                'blocksize','auto', ...
                'tlist',[colon(0,delay,delay)], ...
                'rtol',num2str(rtol), ...
                'tout','tsteps', ...    %Output solution at points used for time integrator
                'tsteps','strict', ...   %Set to strict means it will have to step through each point in tout.
                 'maxorder',3, ...        %Maximum used BDF order. Default Minimum is one which it starts with.
                 'atol',{num2str(atol)}, ...
                'linsolver','gmres',...
                'initialstep',initialstep, ...
              'prepar',{'droptol',droptol});
%%%%%%%
         fem0=fem;
          flclear fem;
 %%%%%%%
 t=fem0.sol.tlist;
 u=postinterp(fem0,'u',[-.5-epsilon;0;0],'solnum',linspace(1,length(fem0.sol.tlist),length(fem0.sol.tlist)));
 v=postinterp(fem0,'v',[.5+epsilon;0;0],'solnum',linspace(1,length(fem0.sol.tlist),length(fem0.sol.tlist)));
 uin=postinterp(fem0,'u',[.5+epsilon;0;0],'solnum',linspace(1,length(fem0.sol.tlist),length(fem0.sol.tlist)));
 vin=postinterp(fem0,'v',[-.5-epsilon;0;0],'solnum',linspace(1,length(fem0.sol.tlist),length(fem0.sol.tlist)));
%%Create text files which stores uin and vin (values on their respective compartments)
fid = fopen('uin.txt', 'wt');
fprintf(fid,'%.20f %.20f  \n',[t;uin']);
fclose(fid);
fid = fopen('vin.txt', 'wt');
fprintf(fid,'%.20f %.20f  \n',[t;vin']);
 fclose(fid);
%%Create text files which stores u and v on their opposite compartments.
fid = fopen('u.txt', 'wt');
fprintf(fid,'%.20f %.20f  \n',[t;u']);
fclose(fid);
fid = fopen('v.txt', 'wt');
fprintf(fid,'%.20f %.20f  \n',[t;v']);
 fclose(fid);
%Post Processing
 time=t;
 usol=postinterp(fem0,'u',[-1;0;0],'solnum',linspace(1,length(fem0.sol.tlist),length(fem0.sol.tlist)));
 vsol=postinterp(fem0,'v',[1;0;0],'solnum',linspace(1,length(fem0.sol.tlist),length(fem0.sol.tlist)));
 usolin=uin;
 vsolin=vin;
 %%% LOOP
for j=1:1:N
tic
% Constants
fem.const = {'R','1', ...
  'k1',num2str(k1), ...
  'k2',num2str(k2), ...
  'Du',num2str(Du), ...
  'Dv',num2str(Dv), ...
  'epsilon',num2str(epsilon), ...
  'ku','k1*epsilon', ...
  'kv','k2*epsilon', ...
  'tauu','R^2/Du', ...
  'tauv','R^2/Dv', ...
  'alphau','sqrt(ku/Du)*R', ...
```

```
  'alphav','sqrt(kv/Dv)*R'};
% Geometry
g1=sphere3('1','pos',{'0','0','0'},'axis',{'0','0','1'},'rot','0','const',fem.const);
g2=sphere3('epsilon','pos',{'-0.5','0','0'},'axis',{'0','0','1'},'rot','0','const',fem.const);
g3=sphere3('epsilon','pos',{'0.5','0','0'},'axis',{'0','0','1'},'rot','0','const',fem.const);
g4=geomcomp({g1,g2,g3},'ns',{'g1','g2','g3'},'sf','g1-g2-g3','face','none','edge','all');
% Analyzed geometry
clear s
s.objs={g4};
s.name={'CO1'};
s.tags={'g4'};
fem.draw=struct('s',s);
fem.geom=geomcsg(fem);
% Initialize mesh
fem.mesh=meshinit(fem, ...
                  'hauto',5);
% (Default values are not included)
% Functions
clear fcns
fcns{1}.type='interp';
fcns{1}.name='F';
fcns{1}.filename=[pwd '/u.txt'];
fcns{1}.fileindex='1';
fcns{1}.method='cubicspline';
fcns{1}.extmethod='interior';
fcns{2}.type='interp';
fcns{2}.name='G';
fcns{2}.filename=[pwd '/v.txt'];
fcns{2}.fileindex='1';
fcns{2}.method='cubicspline';
fcns{2}.extmethod='interior';
fcns{3}.type='interp';
fcns{3}.name='Fin';
fcns{3}.filename=[pwd '/uin.txt'];
fcns{3}.fileindex='1';
fcns{3}.method='cubicspline';
fcns{3}.extmethod='interior';
fcns{4}.type='interp';
fcns{4}.name='Gin';
fcns{4}.filename=[pwd '/vin.txt'];
fcns{4}.fileindex='1';
fcns{4}.method='cubicspline';
fcns{4}.extmethod='interior';
fem.functions = fcns;
% Application mode 1
clear appl
appl.mode.class = 'FlPDEC';
appl.dim = {'u','v','u_t','v_t'};
appl.gporder = 4;
appl.cporder = 2;
appl.sshape = 2;
appl.assignsuffix = '_c';
clear bnd
bnd.g = {0,{['1/epsilon*' Fuvloop];0},{0;['1/epsilon*' Guvloop]}};
bnd.name = {'outer','uin','vin'};
bnd.type = 'neu';
bnd.ind = [1,1,1,1,3,3,3,3,3,3,3,1,1,1,1,2,2,2,2,2,2,2,2];
appl.bnd = bnd;
clear equ
equ.f = 0;
equ.da = {{'tauu';'tauv'}};
equ.c = {{'1';'1'}};
equ.a = {{'alphau^2';'alphav^2'}};
equ.init = {{'uinitial(x,y,z)';'vinitial(x,y,z)';0;0}};
equ.ind = [1];
appl.equ = equ;
```

```
fem.appl{1} = appl;
fem.frame = {'ref'};
fem.border = 1;
clear units;
units.basesystem = 'SI';
fem.units = units;
% ODE Settings
clear ode
clear units;
units.basesystem = 'SI';
ode.units = units;
fem.ode=ode;
% Multiphysics
fem=multiphysics(fem);
% Extend mesh
fem.xmesh=meshextend(fem);
         % Solve problem
fem.sol=femtime(fem, ...
                 'init',fem0.sol, ...
                'solcomp',{'v','u'}, ...
                'outcomp',{'v','u'}, ...
                'blocksize','auto', ...
                'tlist',[colon(0,delay,delay)], ...
                'rtol',num2str(rtol), ...
                'tout','tsteps', ...    %Output solution at points used for time integrator
                'tsteps','strict', ...   %Set to strict means it will have to step through each point in tout.
                 'maxorder',3, ...        %Maximum used BDF order
                 'atol',{num2str(atol)}, ...
                'linsolver','gmres',...
                'initialstep',initialstep, ...
              'prepar',{'droptol',droptol});

flclear fem0;
fem0=fem;
flclear fem;
clear t u v uin vin;
t=fem0.sol.tlist;
 u=postinterp(fem0,'u',[-.5-epsilon;0;0],'solnum',linspace(1,length(fem0.sol.tlist),length(fem0.sol.tlist)));
 v=postinterp(fem0,'v',[0.5+epsilon;0;0],'solnum',linspace(1,length(fem0.sol.tlist),length(fem0.sol.tlist)));
 uin=postinterp(fem0,'u',[.5+epsilon;0;0],'solnum',linspace(1,length(fem0.sol.tlist),length(fem0.sol.tlist)));
 vin=postinterp(fem0,'v',[-.5-epsilon;0;0],'solnum',linspace(1,length(fem0.sol.tlist),length(fem0.sol.tlist)));
 %%Create text files which stores uin and vin (values on compartments)
fid = fopen('uin.txt', 'wt');
fprintf(fid,'%.20f %.20f  \n',[t;uin']);
fclose(fid);
fid = fopen('vin.txt', 'wt');
fprintf(fid,'%.20f %.20f  \n',[t;vin']);
 fclose(fid);
%%Create text files which stores u and v on opposing compartments
fid = fopen('u.txt', 'wt');
fprintf(fid,'%.20f %.20f  \n',[t;u']);
fclose(fid);
fid = fopen('v.txt', 'wt');
fprintf(fid,'%.20f %.20f  \n',[t;v']);
 fclose(fid);
timenew=t(2:end)+j*delay;
usolnew=postinterp(fem0,'u',[-1;0;0],'solnum',linspace(2,length(fem0.sol.tlist),length(fem0.sol.tlist)-1));
vsolnew=postinterp(fem0,'v',[1;0;0],'solnum',linspace(2,length(fem0.sol.tlist),length(fem0.sol.tlist)-1));
usolinnew=postinterp(fem0,'u',[.5+epsilon;0;0],'solnum',linspace(2,length(fem0.sol.tlist),length(fem0.sol.tlist)-1));
vsolinnew=postinterp(fem0,'v',[-.5-epsilon;0;0],'solnum',linspace(2,length(fem0.sol.tlist),length(fem0.sol.tlist)-1));
 %Post Processing
time=[time timenew];
usol=[usol; usolnew];
%vsol1=[vsol1; postinterp(fem0,'v',[-1;0;0],'solnum',linspace(2,length(fem0.sol.tlist),length(fem0.sol.tlist)-1))];
vsol=[vsol; vsolnew];
usolin=[usolin; usolinnew];
```

```
vsolin=[vsolin; vsolinnew];
 %%Create text files with time,usol,vsol,usolin,vsolin
fid = fopen('usolin.txt', 'wt');
fprintf(fid,'%.20f %.20f  \n',[time;usolin']);
fclose(fid);
fid = fopen('vsolin.txt', 'wt');
fprintf(fid,'%.20f %.20f  \n',[time;vsolin']);
 fclose(fid);
 fid = fopen('usol.txt', 'wt');
fprintf(fid,'%.20f %.20f  \n',[time;usol']);
 fclose(fid);
 fid = fopen('vsol.txt', 'wt');
fprintf(fid,'%.20f %.20f  \n',[time;vsol']);
 fclose(fid);
toc
end
```

A.4 Solving Delay PDE in (3.2) with Reverse Method of Lines

This is the code described in §3.4.2. It is not finished and is in a preliminary state. It only accepts a constant step size and this step size can not be taken too small. We also have code like this which uses the BDF2 method with a constant step size and the BDF2 method with a variable step size. They are also very preliminary states.

```
% COMSOL Multiphysics Model M-file
% Generated by COMSOL 3.5a (COMSOL 3.5.0.608, $Date: 2009/05/11 07:38:49 $)
tic
load coordvectors001.mat   %This file contains a bunch of (x,y,z) points in a
                            %sphere of radius 1.Stored in a structure called pd. The x,y,z data
                            %came from COMSOL GUI and does not include points
                            %inside the compartments because they are not in domain.
                            %This struct is very large and the line below this can
                            %be used several times to use less points in the domain
                            %for the interpolation.
% pd.x(2:2:end)=[];  pd.y(2:2:end)=[];  pd.z(2:2:end)=[];
flclear fem; clear fem;
flclear fem0; clear fem0;
% COMSOL version
clear vrsn
vrsn.name = 'COMSOL 3.5';
vrsn.ext = 'a';
vrsn.major = 0;
vrsn.build = 608;
vrsn.rcs = '$Name: v35ap $';
vrsn.date = '$Date: 2009/05/11 07:38:49 $';
fem.version = vrsn;
initu=1;  %This is the value used for the constant history
           %data for u  defined for -delay<=t<0
initv=1;   %This is the value used for the constant
            %history data for u  defined for -delay<=t<0
epsilon=0.01;
delay=1.5/epsilon;
ts=2;     %time step
timesteps=2350;
delaysuout=zeros(delay/ts,1); %History Data: Holds
                              %solution of u on
                              %compartment which is assumed to be constant.
```

```
delaysuin=zeros(delay/ts,1);
delaysvout=zeros(delay/ts,1);
delaysvin=zeros(delay/ts,1);
delaysuout(1:delay/ts)=initu;
delaysuin(1:delay/ts)=initu;
delaysvout(1:delay/ts)=initv;
delaysvin(1:delay/ts)=initv;
uxsolout=zeros(1,timesteps+1);      %number of rows is the number of
                                    %points to plot solution on x-axis
                                    %between -1 and 1
usolin=zeros(1,timesteps+1);
vsolin=zeros(1,timesteps+1);
vxsolout=zeros(1,timesteps+1);
%Know solution at t=0
uxsolout(1)=delaysuout(1);
vxsolout(1)=delaysvout(1);
usolin(1)=delaysuout(1);
vsolin(1)=delaysvout(1);
%Create Fem struct and solve the first
%iteration which gives solution at time t=ts
% Constants
fem.const = {'R','1', ...
  'ts',num2str(ts), ...   %time step
  'k1','1/2', ...
  'k2','1', ...
  'Du','1/3', ...
  'Dv','1/3', ...
  'epsilon',num2str(epsilon), ...
   'ku','k1*epsilon', ...
  'kv','k2*epsilon', ...
  'tauu','R^2/Du', ...
  'tauv','R^2/Dv', ...
  'alphau','sqrt(ku/Du)*R', ...
  'alphav','sqrt(kv/Dv)*R'};
% Geometry
g1=sphere3('1','pos',{'0','0','0'},'axis', {'0','0','1'},'rot','0');
g2=sphere3(epsilon,'pos',{'1/2','0','0'}, 'axis',{'0','0','1'},'rot','0');
g3=sphere3(epsilon,'pos',{'-1/2','0','0'}, 'axis',{'0','0','1'},'rot','0');
g4=geomcomp({g1,g2,g3},'ns',{'g1','g2','g3'},'sf','g1-g2-g3','face','none','edge','all');
% Analyzed geometry
clear s
s.objs={g4};
s.name={'CO1'};
s.tags={'g4'};
fem.draw=struct('s',s);
fem.geom=geomcsg(fem);
% Initialize mesh
fem.mesh=meshinit(fem, 'hauto',5);
% % Refine mesh
% fem.mesh=meshrefine(fem, 'mcase',0, 'rmethod','longest');
% Application mode 1
clear appl
appl.mode.class = 'FlPDEC';
appl.dim = {'u','v','u_t','v_t'};
appl.gporder = 4;
appl.cporder = 2;
appl.sshape = 2;
appl.assignsuffix = '_c';
clear bnd
bnd.g = {0,{1/epsilon*ts*fbc(delaysuin(delay/ts),delaysvout(delay/ts));0},{0;1/epsilon*ts*gbc(delaysuout(delay/ts),...
delaysvin(delay/ts))}};
bnd.name = {'out','uin','vin'};
bnd.type = 'neu';
bnd.ind = [1,1,1,1,3,3,3,3,3,3,3,3,1,1,1,1,2,2,2,2,2,2,2,2];
appl.bnd = bnd;
clear equ
```

```
equ.f = {{['tauu*' num2str(delaysuout(1))];['tauv*'num2str(delaysvout(1))];}};
equ.da = {{1;1}};
equ.c = {{'ts';'ts'}};
equ.a = {{'alphau^2*ts+tauu';'alphav^2*ts+tauv'}};
equ.init = {{delaysuout(1);delaysvout(1);0;0}};
equ.ind = [1];
appl.equ = equ;
fem.appl{1} = appl;
fem.frame = {'ref'};
fem.border = 1;
clear units;
units.basesystem = 'SI';
fem.units = units;
% ODE Settings
clear ode
clear units;
units.basesystem = 'SI';
ode.units = units;
fem.ode=ode;
% Multiphysics
fem=multiphysics(fem);
% Extend mesh
fem.xmesh=meshextend(fem);
%Solve problem with fixed spacial mesh
fem.sol=femstatic(fem, ...
                  'solcomp',{'v','u'}, ...
                  'outcomp',{'v','u'}, ...
                  'blocksize','auto', ...
                  'ntol',1.0E-6, ...
                  'maxiter',25, ...
                  'linsolver','gmres', ...
                  'prepar',{'droptol',0.01});
delaysuout=delaysuout([ end 1:end-1 ]);
delaysuin=delaysuin([ end 1:end-1 ]);
delaysvout=delaysvout([ end 1:end-1 ]);
delaysvin=delaysvin([ end 1:end-1 ]);
delaysuout(1)=postinterp(fem,'u',[-.5-epsilon;0;0]);
delaysvout(1)=postinterp(fem,'v',[0.5+epsilon;0;0]);
delaysuin(1)=postinterp(fem,'u',[1/2+epsilon;0;0]);
delaysvin(1)=postinterp(fem,'v',[-1/2-epsilon;0;0]);
uxsolout(2)=postinterp(fem,'u',[-1;0;0])';
usolin(2)=postinterp(fem,'u',[.5+epsilon;0;0])';
vsolin(2)=postinterp(fem,'v',[-.5-epsilon;0;0])';
vxsolout(2)=postinterp(fem,'v',[1;0;0])';
fem0=fem;   %Store the Fem from first iteration (time=ts) in fem0.
 for jj = 2:timesteps
toc
tic
% Constants
fem.const = {'R','1', ...
  'ts',num2str(ts), ...   %time step
  'k1','1/2', ...
  'k2','1', ...
  'Du','1/3', ...
  'Dv','1/3', ...
  'epsilon',num2str(epsilon), ...
   'ku','k1*epsilon', ...
  'kv','k2*epsilon', ...
  'tauu','R^2/Du', ...
  'tauv','R^2/Dv', ...
  'alphau','sqrt(ku/Du)*R', ...
  'alphav','sqrt(kv/Dv)*R'};
% Geometry
g1=sphere3('1','pos',{'0','0','0'},'axis', {'0','0','1'},'rot','0');
g2=sphere3(epsilon,'pos',{'1/2','0','0'}, 'axis',{'0','0','1'},'rot','0');
g3=sphere3(epsilon,'pos',{'-1/2','0','0'}, 'axis',{'0','0','1'},'rot','0');
```

```
g4=geomcomp({g1,g2,g3},'ns',{'g1','g2','g3'}, 'sf','g1-g2-g3','face','none','edge','all');

% Analyzed geometry
clear s
s.objs={g4};
s.name={'CO1'};
s.tags={'g4'};

fem.draw=struct('s',s);
fem.geom=geomcsg(fem);

% Initialize mesh
fem.mesh=meshinit(fem, 'hauto',5);
%Create Interpolation text file which is used to compute interpolation
%function
clear datauu datavv;
[datauu] = postinterp(fem0,'u',[pd.x;pd.y;pd.z]);
fid = fopen('datau.txt', 'wt');
fprintf(fid,'%13f %13f %13f %13f \n',[pd.x;pd.y;pd.z;datauu]);
fclose(fid);
%Create Interpolation text file which is used to compute interpolation
%function
[datavv] = postinterp(fem0,'v',[pd.x;pd.y;pd.z]);
fid = fopen('datav.txt', 'wt');
fprintf(fid,'%13f %13f %13f %13f \n',[pd.x;pd.y;pd.z;datavv]);
fclose(fid);
clear fcns
fcns{1}.type='interp';
fcns{1}.name='datau';
fcns{1}.filename='/home/clevy/phd/Model With Delay/implicit euler/datau.txt';
fcns{1}.fileindex='1';
fcns{1}.method='linear';
fcns{1}.extmethod='const';
fcns{1}.defvars='true';
fcns{2}.type='interp';
fcns{2}.name='datav';
fcns{2}.filename='/home/clevy/phd/Model With Delay/implicit euler/datav.txt';
fcns{2}.fileindex='1';
fcns{2}.method='linear';
fcns{2}.extmethod='const';
fcns{2}.defvars='true';
fem.functions = fcns;
% Application mode 1
clear appl
appl.mode.class = 'FlPDEC';
appl.dim = {'u','v','u_t','v_t'};
appl.gporder = 4;
appl.cporder = 2;
appl.sshape = 2;
appl.assignsuffix = '_c';
clear bnd
bnd.g = {0,{1/epsilon*ts*fbc(delaysuin(delay/ts),delaysvout(delay/ts));0}, {0;1/epsilon*ts*gbc(delaysuout(delay/ts),...
delaysvin(delay/ts))}};
bnd.name = {'out','uin','vin'};
bnd.type = 'neu';
bnd.ind = [1,1,1,1,3,3,3,3,3,3,3,1,1,1,1,2,2,2,2,2,2,2,2];
appl.bnd = bnd;
clear equ
equ.f = {{'tauu*datau(x,y,z)';'tauv*datav(x,y,z)'}};
equ.da = {{1;1}};
equ.c = {{ts;ts}};
equ.a = {{'alphau^2*ts+tauu';'alphav^2*ts+tauv'}};
equ.init = {{'datau(x,y,z)';'datav(x,y,z)';0;0}};    %Initial Guess
equ.ind = [1];
appl.equ = equ;
fem.appl{1} = appl;
```

```
fem.frame = {'ref'};
fem.border = 1;
clear units;
units.basesystem = 'SI';
fem.units = units;
% ODE Settings
clear ode
clear units;
units.basesystem = 'SI';
ode.units = units;
fem.ode=ode;
% Multiphysics
fem=multiphysics(fem);
% Extend mesh
fem.xmesh=meshextend(fem);
% Solve problem with fixed spacial mesh
fem.sol=femstatic(fem, ...
                  'solcomp',{'v','u'}, ...
                  'outcomp',{'v','u'}, ...
                  'blocksize','auto', ...
                  'ntol',1.0E-6, ...
                  'maxiter',25, ...
                  'linsolver','gmres', ...
                  'prepar',{'droptol',0.01});
delaysuout=delaysuout([ end 1:end-1 ]);
delaysvout=delaysvout([ end 1:end-1 ]);
delaysuout(1)=postinterp(fem,'u',[-.5-epsilon;0;0]);
delaysvout(1)=postinterp(fem,'v',[.5+epsilon;0;0]);
delaysuin=delaysuin([ end 1:end-1 ]);
delaysvin=delaysvin([ end 1:end-1 ]);
delaysuin(1)=postinterp(fem,'u',[1/2+epsilon;0;0]);
delaysvin(1)=postinterp(fem,'v',[-1/2-epsilon;0;0]);
uxsolout(jj+1)=postinterp(fem,'u',[-1;0;0])';
vxsolout(jj+1)=postinterp(fem,'v',[1;0;0])';
usolin(jj+1)=postinterp(fem,'u',[.5+epsilon;0;0]);
vsolin(jj+1)=postinterp(fem,'v',[-.5-epsilon;0;0]);
% %With this you can look at the accuracy of the interpolation function. If
% %these two plots are on top of eachother than that is good.
% close
% figure flclear fem0; clear fem0;
fem0=fem;
delete('datau.txt');
delete('datav.txt');
jj
 end
```

A.5 Calculating ω_2 as described in §3.6

A.5.1 Script to Calculate ω_2

This script calculates ω_2 as defined in (3.53). It relies on all the other scripts and functions that are found in the following subsections. The code here in §A.5 was used for the example in §3.6.2.

```
%This code works for Du=1/3 and Dv=1/3 and R=1
%This code calculates omega2 for the Hopf in the Delay Model to get rid of
%secular growth.
```

```
clear all; close all; clc;
delay=2;
k1=1;
k2=1;
omega1=1;  %This is always set equal to one for this code.
           %Decided not to rescale the period to 2*pi
R11=-0.8553180758e-1;   %These 4 values here are the  Green's function values. If the coordinates of x1 and %x2  are
                          %(0.5,0,0) and (-0.5,0,0)then they can stay like this.
R22=-0.8553180758e-1;
GN21=-0.6817258485e-1;
GN12=-0.6817258485e-1;
uhist=linspace(.5,.5,1000);   %These 8 vectors set the history for these functions
vhist=linspace(.8,.8,1000);
c1hist=linspace(0,0,1000);
c2hist=linspace(0,0,1000);
chi1hist=linspace(0,0,1000);
chi2hist=linspace(0,0,1000);
A1hist=linspace(0,0,1000);
A2hist=linspace(0,0,1000);
thist=linspace(-delay,0,length(uhist));
save methstepshistory.mat delay k1 k2 omega1 R11 R22 GN21 GN12 thist uhist vhist c1hist c2hist A1hist A2hist ...
 chi1hist chi2hist
clear
%Run Method Steps to get periodic orbit at end
N=60;  %Number of delay intervals to solve on.
%Make this large enough to get rid of transient
%and get periodic solution
run_method_steps_leading_order
%Get the new history for the next calculation. This will get rid of
%transient.
uhist=spline(t,u,linspace(t(end)-delay,t(end),1000));
vhist=spline(t,v,linspace(t(end)-delay,t(end),1000));
c1hist=spline(t,c1,linspace(t(end)-delay,t(end),1000));
c2hist=spline(t,c2,linspace(t(end)-delay,t(end),1000));
save methstepshistory.mat delay k1 k2 omega1 R11 R22 GN21 GN12 thist uhist vhist c1hist c2hist A1hist A2hist ...
 chi1hist chi2hist
clear

%Run Method of Steps to get atleast one period no transient
N=30;%Number of delay intervals to solve on.
%Make this large enough to get a few periods
run_method_steps_leading_order
uasym=u;
%get the period with get_periods_better
get_periods_better  %this calculates period
ttp=linspace(0,period,1000);
u=spline(t,u,ttp);
v=spline(t,v,ttp);
c1=spline(t,c1,ttp);
c2=spline(t,c2,ttp);
save lead_order_system.mat u v c1 c2 ttp
clearvars -except period
load methstepshistory.mat
save methstepshistory.mat
clear
%Now do the runeigsc.m and make sure the omega 2 converges.
%Do this by checking the Omega2s array
count=1;
for n=100:100:400  %h=period/(n+1) is the time step in the discretization.
runeigsc    %Increase n for more accuracy. It leads to finding the eigenvalues of a larger matrix
Omega2s(count)=omega2;
H(count)=h;
N(count)=n;
count=count+1;
clearvars -except  n Omega2s H count N
end
```

```
%Check Omega2s for convergence in omega2

omega2=Omega2s(end);
clearvars -except omega2
load methstepshistory.mat
save methstepshistory.mat
%Then you can run method of steps the whole system  When running the method of steps with the correct
%omeaga2 you may want to change the history back to constants for u,v,c1,c2
uhist=linspace(.5,.5,1000);   %These 8 vectors
%set the history for these functions
vhist=linspace(.8,.8,1000);
c1hist=linspace(0,0,1000);
c2hist=linspace(0,0,1000);
chi1hist=linspace(0,0,1000);
chi2hist=linspace(0,0,1000);
A1hist=linspace(0,0,1000);
A2hist=linspace(0,0,1000);
thist=linspace(-delay,0,length(uhist));
save methstepshistory.mat
clear
%Now run method of steps to solve the two DDAE systems for
%u,v,c1,c1,chi1,chi2,A1,A2
load methstepshistory.mat;
N=100;
run_method_steps
```

A.5.2 Script to Solve Leading Order DDAEs in (3.34)

This code solves the leading order DDAEs found in (3.34).

```
%This assumes tauu=tauv=3  which is brought about by setting R=1,Du=1/3=Dv
load methstepshistory.mat %This loads thist,uhist,vhist,c1hist,c2hist,chi1hist,chi2hist,A1hist,A2hist
%It also loads among other things. See the file if interested.
m=100;  %delay/m is the step size in tt
%Options for ODE solver
options=odeset('stats','off','RelTol', 1e-9,'AbsTol',1e-9,'InitialStep',1e-9,'Events',@(t,y) ...
 event_function(t,y,delay));

%These history vectors are always defined on [-delay,0] throughout this code. We solve each
%step using ode45 on [0,delay] which uses the history from [-delay,0].
%Define the history for the system on [0,delay]
%Note that thist=[-delay,0] and so does th
th=linspace(-delay,0,m+1);                  t=th;
c1h=spline(thist,c1hist,th);                c1=c1h;
c2h=spline(thist,c2hist,th);                c2=c2h;
uh=spline(thist,uhist,th);                  u=uh;
vh=spline(thist,vhist,th);                  v=vh;
%Solve the system on [0,delay]
sol=ode45(@(t,y) odesys_leading_order(t,y,th,uh,vh,c1h,c2h,omega1,delay,k1,k2),[0 delay],[uh(end) vh(end)],options);
yn=deval(sol,th+delay);
for j=1:N
%History for the system on [j*delay,(j+1)delay]
th=linspace(-delay,0,m+1); t=[t(1:end-1) th+j*delay];
c1h=f(uh+c1h,vh);       c1=[c1(1:end-1) c1h];
c2h=g(uh,vh+c2h);       c2=[c2(1:end-1) c2h];
uh=yn(1,:);             u=[u(1:end-1) uh];
vh=yn(2,:);             v=[v(1:end-1) vh];
%Solve the system on [j*delay,(j+1)*delay]
%if j==10
 %   options=odeset('stats','off','RelTol',
 %1e-3,'AbsTol',1e-3,'InitialStep',1e-1,'Events',@(t,y)
```

```
 %event_function(t,y,delay));
%end
sol=ode45(@(t,y) odesys_leading_order(t,y,th,uh,vh,c1h,c2h,omega1,delay,k1,k2),[0 delay],[uh(end) vh(end)],options);
yn=deval(sol,th+delay);
j
end
```

This code defines the leading order DDAE system in (3.34).

```
function dydt = odesys_leading_order(t,y,th,uh,vh,c1h,c2h,omega1,delay,k1,k2)
%This assumes tauu=tauv=3  which is brought about by setting R=1,
%Du=1/3=Dv=1/3
%reason for t-delay is that th is on [-delay,0] and t is on [0,delay]
F=spline(th,f(uh+c1h,vh),t-delay);
G=spline(th,g(uh,vh+c2h),t-delay);
dydt =1/omega1*[-k1*y(1)+F
    -k2*y(2)+G];
```

A.5.3 Script to Solve the DDAEs in (3.42) and (3.47)

This script can be used to solve the DDAEs defined by (3.46) and (3.47). If we set $\omega_2 = 0$ in this code then it also can solve the DDAEs in (3.42).

```
%This assumes tauu=tauv=3  which is brought about by setting R=1,
%Du=1/3=Dv

tic
m=100;  %delay/m is the step size in tt (just post processing)
%Options for ODE solver
options=odeset('stats','off','RelTol', 1e-7,'AbsTol',1e-7,...
    'InitialStep',1e-7,'Events',@(t,y) event_function(t,y,delay));
%These history vectors are always defined on [-delay,0] throughout this
%code. We solve each step using ode45 on [0,delay] which uses the history
%from [-delay,0].
%Define the history for the system on [0,delay]
%Note that thist=[-delay,0] and so does th
%The variables defined on the right (or secondly) are for post processing
%only.
th=linspace(-delay,0,m+1);                  t=th;
A1h=spline(thist,A1hist,th);                A1=A1h;
A2h=spline(thist,A2hist,th);                A2=A2h;
c1h=spline(thist,c1hist,th);                c1=c1h;
c2h=spline(thist,c2hist,th);                c2=c2h;
chi1h=spline(thist,chi1hist,th);           chi1=chi1h;
chi2h=spline(thist,chi2hist,th);            chi2=chi2h;
uh=spline(thist,uhist,th);                  u=uh;
vh=spline(thist,vhist,th);                  v=vh;
uph=ppval(fnder(spline(th,uh)),th);
vph=ppval(fnder(spline(th,vh)),th);
c1ph=ppval(fnder(spline(th,c1h)),th);
c2ph=ppval(fnder(spline(th,c2h)),th);
%Solve the system on [0,delay]
sol=ode45(@(t,y) odesys(t,y,omega2,th,uh,vh,c1h,c2h,chi1h,chi2h,A1h,A2h,omega1,delay,k1,k2,R11,R22,GN21,GN12),...
[0 delay],[uh(end) vh(end) chi1h(end) chi2h(end)],options);
yn=deval(sol,th+delay);
for j=1:N
%History for the system on [j*delay,(j+1)delay]
```

```
th=linspace(-delay,0,m+1); t=[t(1:end-1) th+j*delay];
A1h=fu(uh+c1h,vh).*(4*pi*c1h*R11+chi1h+A1h)+fv(uh+c1h,vh).*(4*pi*c2h*GN12+chi2h)-delay/omega1*omega2*...
(fu(uh+c1h,vh).*(uph+c1ph)+fv(uh+c1h,vh).*vph); A1=[A1(1:end-1) A1h];
A2h=gv(uh,vh+c2h).*(4*pi*c2h*R22+chi2h+A2h)+gu(uh,vh+c2h).*(4*pi*c1h*GN21+chi1h)-delay/omega1*omega2*...
(gv(uh,vh+c2h).*(vph+c2ph)+gu(uh,vh+c2h).*uph); A2=[A2(1:end-1) A2h];
c1h=f(uh+c1h,vh);       c1=[c1(1:end-1) c1h];
c2h=g(uh,vh+c2h);       c2=[c2(1:end-1) c2h];
chi1h=yn(3,:);          chi1=[chi1(1:end-1) chi1h];
chi2h=yn(4,:);          chi2=[chi2(1:end-1) chi2h];
uh=yn(1,:);             u=[u(1:end-1) uh];
vh=yn(2,:);             v=[v(1:end-1) vh];
uph=ppval(fnder(spline(th,uh)),th);
vph=ppval(fnder(spline(th,vh)),th);
c1ph=ppval(fnder(spline(th,c1h)),th);
c2ph=ppval(fnder(spline(th,c2h)),th);
%Solve the system on [j*delay,(j+1)*delay]
%if j==10
 %   options=odeset('stats','off','RelTol', 1e-3,'AbsTol',1e-3,
 %'InitialStep',1e-1,'Events',@(t,y) event_function(t,y,delay));
%end
sol=ode45(@(t,y) odesys(t,y,omega2,th,uh,vh,c1h,c2h,chi1h,chi2h, A1h,A2h,omega1,delay,k1,k2,R11,R22,GN21,...
GN12),[0 delay],[uh(end) vh(end) chi1h(end) chi2h(end)],options);
yn=deval(sol,th+delay);
j
end
toc
```

This code defines the DDAE system by (3.46) and (3.47). Setting $\omega_2 = 0$, it also defines the DDAE system in (3.42).

```
function dydt = odesys(t,y,omega2,th,uh,vh,c1h,c2h,chi1h,...
    chi2h,A1h,A2h,omega1,delay,k1,k2,R11,R22,GN21,GN12)
%This assumes tauu=tauv=3  which is brought about by setting R=1,
%Du=1/3=Dv=1/3
%reason for t-delay is that th is on [-delay,0] and t is on [0,delay]
F=spline(th,f(uh+c1h,vh),t-delay);
FU=spline(th,fu(uh+c1h,vh),t-delay);
FV=spline(th,fv(uh+c1h,vh),t-delay);
G=spline(th,g(uh,vh+c2h),t-delay);
GU=spline(th,gu(uh,vh+c2h),t-delay);
GV=spline(th,gv(uh,vh+c2h),t-delay);
A1=spline(th,A1h,t-delay);
A2=spline(th,A2h,t-delay);
c1=spline(th,c1h,t-delay);
c2=spline(th,c2h,t-delay);
chi1=spline(th,chi1h,t-delay);
chi2=spline(th,chi2h,t-delay);
up=ppval(fnder(spline(th,uh)),t-delay);
vp=ppval(fnder(spline(th,vh)),t-delay);
c1p=ppval(fnder(spline(th,c1h)),t-delay);
c2p=ppval(fnder(spline(th,c2h)),t-delay);
dydt =1/omega1*[-k1*y(1)+F
    -k2*y(2)+G
    -k1*y(3)-omega2/omega1*(-k1*y(1)+F)+FU*(4*pi*c1*R11+chi1+A1)+FV*(4*pi*c2*GN12+chi2)-delay/omega1*omega2*...
(FU*(up+c1p)+FV*vp)
    -k2*y(4)-omega2/omega1*(-k2*y(2)+G)+GV*(4*pi*c2*R22+chi2+A2)+GU*(4*pi*c1*GN21+chi1)-delay/omega1*omega2*...
(GV*(vp+c2p)+GU*up)];
```

These functions define $F(u, v)$ and $G(u, v)$ and their partial derivatives.

```
function s = f(u,v)
s=4*u./((1/2+u).*(1+v.^2));
function s = fu(u,v)
s=8./((1+2*u).^2.*(1+v.^2));
function s = fv(u,v)
s=-16*u.*v./((1+2*u).*(1+v.^2).^2);
function s = g(u,v)
s=2*v.*u./(1/4+v);
function s = gu(u,v)
s=8*v./(4*v+1);
function s = gv(u,v)
s=8*u./(1+4*v).^2;
```

This event function is used in the ODE solvers that are used in the method steps to solve the DDAE systems. This event function prevents the solver from stepping through a discontinuity and then interpolating back over it.

```
function [value,isterminal,direction] = event_function(t,y,delaypoints)
% when value is equal to zero, an event is triggered.
% set isterminal to 1 to stop the solver at the first event, or 0 to
% get all the events.
%  direction=0 if all zeros are to be computed (the default), +1 if
%  only zeros where the event function is increasing, and -1 if only
%  zeros where the event function is decreasing.
value = t-delaypoints;  % when value = 0, an event is triggered
%value = t-t;  % when value = 0, an event is triggered
isterminal = 1; % terminate after the first event
direction = 0;  % get all the zeros
```

A.5.4 Finding Period of the Periodic Solutions from (3.46)

This function finds the period of the solutions which satisfy (3.46) in the case of a Hopf bifurcation.

```
%This finds the period of the function whose independent variable is t and whose dependent variable is uasym
% The function is defined by basically spline(t,uasym) The polyfit was illconditioned for these functions
%so we had to a change of variables.A qudratic is fit between two adjacent peaks and then
%period is the difference between the peaks
 [pks,locs]=findpeaks(uasym);
lp1=locs(1)-1;
rp1=locs(1)+1;
lp2=locs(2)-1;
rp2=locs(2)+1;
V=uasym;
[Y1,I1]=max(V(lp1:rp1));
t1=t(lp1+I1-2);  x1=uasym(lp1+I1-2);
t2=t(lp1+I1-1);  x2=uasym(lp1+I1-1);
t3=t(lp1+I1);    x3=uasym(lp1+I1);
tlist=[t1 t2 t3];
tnew = (tlist-mean(tlist))/std(tlist);
[p,S,mu] = polyfit(tlist,[x1 x2 x3],2);
Anew = p(1); Bnew = p(2); Cnew = p(3); mu1 = mu(1); mu2 = mu(2);
C=(Anew*(mu1)^2/(mu2)^2)-(Bnew*(mu1/mu2))+Cnew;
```

```
B = (Bnew*mu2 -2*Anew*mu1)/(mu2)^2;
A = Anew/(mu2)^2;
t1max=-B/(2*A);
[Y1,I1]=max(V(lp2:rp2));
t1=t(lp2+I1-2);  x1=uasym(lp2+I1-2);
t2=t(lp2+I1-1);  x2=uasym(lp2+I1-1);
t3=t(lp2+I1);    x3=uasym(lp2+I1);
tlist=[t1 t2 t3];
tnew = (tlist-mean(tlist))/std(tlist);
[p,S,mu] = polyfit(tlist,[x1 x2 x3],2);
Anew = p(1); Bnew = p(2); Cnew = p(3); mu1 = mu(1); mu2 = mu(2);
C=(Anew*(mu1)^2/(mu2)^2)-(Bnew*(mu1/mu2))+Cnew;
B = (Bnew*mu2 -2*Anew*mu1)/(mu2)^2;
A = Anew/(mu2)^2;
t2max=-B/(2*A);
period=t2max-t1max;
```

A.5.5 Discretization of the Adjoint Operator

This code creates the discretized operator (matrix), $\overline{\boldsymbol{L}}$, defined in (3.61).

```
function L=eigsc(n,period,delay,k1,k2)
%Y is the vector [Y1, Y2, Y3, Y4] where wach Yj has the form [y0,y1,y2,...yn] where y0=y_{n+1} is the periodic b.c.
%We use finite difference to turn the linear neutral delay operator in to a matrix eigenvalue problem.
% Looking for zero eigenvalues with periodic solutions to use in solvability condition to calculate omega2.
h=period/(n+1);
%This derivative is O(h^4) and is y'~(y_(n-2)-8y_(n-1)+8y_(n+1)-y_(n+2))/(12*h)
%First create one block
dt=zeros(n+1);
dt(1,2)=8; dt(1,3)=-1; dt(1,n)=1;  dt(1,n+1)=-8;  %first row
dt(2,1)=-8; dt(2,3)=8; dt(2,4)=-1; dt(2,n+1)=1;   %second row
dt(n,1)=-1; dt(n,n-2)=1; dt(n,n-1)=-8; dt(n,n+1)=8; %second last row
dt(n+1,1)=8; dt(n+1,2)=-1; dt(n+1,n-1)=1; dt(n+1,n)=-8;  %last row
for j=3:n-1 %all other rows in between
    dt(j,j-2)=1;  dt(j,j-1)=-8; dt(j,j+1)=8; dt(j,j+2)=-1;
end
%Make the Block diagonal derivative matrix
dt=blkdiag(dt,dt,dt,dt);
dt=1/(12*h)*dt;
%Create Omega matrix which is just diag(omega1,omega1,...,1,1,....1)
Omega=diag([linspace(1,1,2*n+2) linspace(0,0,2*n+2)]);
K=diag([linspace(k1,k1,n+1) linspace(k2,k2,n+1) linspace(1,1,2*n+2)]);
%Advancement Matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
frac=delay/h-floor(delay/h);     %the \mu in paper
N=floor(delay/h);
%Advancement Matrix
%Define the R_N matrix in delay paper. Multiplying a vector by this matrix
%shifts every element back N spaces. It brings every point in the
%discretized vector, y_k, to the corresponding y_j in the figure in delay
%paper. See figure/diagram from paper.
rotj=zeros(n+1);
rotj(1:n+1-N,N+1:n+1)=eye(n+1-N);      rotj(n+2-N:n+1,1:N)=eye(N);
%R_(N+1)
N=floor(delay/h)+1;
rotj1=zeros(n+1);
rotj1(1:n+1-N,N+1:n+1)=eye(n+1-N);      rotj1(n+2-N:n+1,1:N)=eye(N);
%R_(N+2)
N=floor(delay/h)+2;
rotj2=zeros(n+1);
rotj2(1:n+1-N,N+1:n+1)=eye(n+1-N);      rotj2(n+2-N:n+1,1:N)=eye(N);
```

```
%R_(N-1)
N=floor(delay/h)-1;
rotjm1=zeros(n+1);
rotjm1(1:n+1-N,N+1:n+1)=eye(n+1-N);      rotjm1(n+2-N:n+1,1:N)=eye(N);
%R_(N-2)
N=floor(delay/h)-2;
rotjm2=zeros(n+1);
rotjm2(1:n+1-N,N+1:n+1)=eye(n+1-N);      rotjm2(n+2-N:n+1,1:N)=eye(N);
%%%%%%%%%%%%%%%%
% %Quartic Approximation
 rot=((1/4)*rotj+(1/24)*rotj2-(1/6)*rotjm1+(1/24)*rotjm2 -(1/6)*rotj1)*frac^4+((1/12)*rotj2+(1/6)*rotjm1-...
(1/12)*rotjm2-(1/6)*rotj1)*frac^3+(-(5/4)*rotj-(1/24)*rotj2+(2/3)*rotjm1- (1/24)*rotjm2+(2/3)*rotj1)*frac^2...
+(-(1/12)*rotj2-(2/3)*rotjm1+ (1/12)*rotjm2+(2/3)*rotj1)*frac+rotj;
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rot=blkdiag(rot,rot,rot,rot);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Load in the Leading order Solution: It loads
%periodic solutions u,v,c1,c2
load lead_order_system.mat;
%Loads ttp,u,v,c1,c2  on [0,period]
%Create the matrix that does the operation F(t)*Y(t) where F(t) is a time depen. matrix.
%F(t) is a 4 by 4 matrix with 16 elements. Each element is a function of
%the form f_{ij}(t) where i=1..4, j=1..4. There will be 16 blocks in the
%correspinding operator Matrix each of size n+1 by n+1
f1=@(T) spline(ttp,fu(u+c1,v),T);  f2=@(T) spline(ttp,fv(u+c1,v),T);    f3=@(T) f1(T);      f4=@(T) 0;
f5=@(T) spline(ttp,gu(u,c2+v),T);   f6=@(T) spline(ttp,gv(u,c2+v),T);  f7=@(T) 0;  f8=@(T) spline(ttp,gv(u,c2+v),T);
f9=@(T) spline(ttp,fu(u+c1,v),T);  f10=@(T) spline(ttp,fv(u+c1,v),T);       f11=@(T) f9(T);       f12=@(T) 0;
f13=@(T) spline(ttp,gu(u,c2+v),T);  f14=@(T) spline(ttp,gv(u,c2+v),T);      f15=@(T) 0;       f16=@(T) f14(T);
F=zeros(size(rot));
for i=1:n+1
F(i,i)=f1((i-1)*h); F(i,i+n+1)=f2((i-1)*h); F(i,i+2*n+2)=f3((i-1)*h); F(i,i+3*n+3)=f4((i-1)*h);
F(i+n+1,i)=f5((i-1)*h); F(i+n+1,i+n+1)=f6((i-1)*h); F(i+n+1,i+2*n+2)=f7((i-1)*h);F(i+n+1,i+3*n+3)=f8((i-1)*h);
F(i+2*n+2,i)=f9((i-1)*h); F(i+2*n+2,i+n+1)=f10((i-1)*h); F(i+2*n+2,i+2*n+2)=f11((i-1)*h);
F(i+2*n+2,i+3*n+3)=f12((i-1)*h);F(i+3*n+3,i)=f13((i-1)*h); F(i+3*n+3,i+n+1)=f14((i-1)*h);
F(i+3*n+3,i+2*n+2)=f15((i-1)*h); F(i+3*n+3,i+3*n+3)=f16((i-1)*h);
end
J2=F;
L=-Omega*dt-J2'*rot+K;
```

A.5.6 Finding the Kernel of the Discretized Operator

This code finds the kernel of $\overline{\boldsymbol{L}}$ which is defined in (3.61). This kernel approximates the kernel of the operator $\mathcal{L}^*$ defined in (3.52). This code also calculates ω_2 from the solvability condition in (3.53).

```
%this caluclates omega2 by finding the kernel of the adjoint. uses eigs to find kernel of the
%discretized operator (a matrix)
load methstepshistory.mat
h=period/(n+1);
L=eigsc(n,period,delay,k1,k2);
[v,e]=eigs(L,1,.00000001);
e=diag(e);
 w1=v(:,1);
t=linspace(0,period-h,n+1);
x1=w1(1:n+1,1);
x2=w1(n+2:2*n+2,1);
x3=w1(2*n+3:3*n+3,1);
x4=w1(3*n+4:4*n+4,1);
%Add in the last point t_{n+1}=Period
```

```
t(end+1)=t(end)+h;
x1(end+1)=x1(1);
x2(end+1)=x2(1);
x3(end+1)=x3(1);
x4(end+1)=x4(1);
timeadj=t;
save adjointsol x1 x2 x3 x4 timeadj
 load lead_order_system.mat %loads ttp,u,v,c1,c2
load adjointsol.mat         %loads
x1=spline(timeadj,x1,ttp);
x2=spline(timeadj,x2,ttp);
x3=spline(timeadj,x3,ttp);
x4=spline(timeadj,x4,ttp);
up=ppval(fnder(spline(ttp,u),1),ttp);
vp=ppval(fnder(spline(ttp,v),1),ttp);
c1p=ppval(fnder(spline(ttp,c1),1),ttp);
c2p=ppval(fnder(spline(ttp,c2),1),ttp);
c1pp=ppval(fnder(spline(ttp,c1),2),ttp);
c2pp=ppval(fnder(spline(ttp,c2),2),ttp);
B1=4*pi*R11*fu(u+c1,v).*c1+4*pi*GN12*fv(u+c1,v).*c2;
B1p=ppval(fnder(spline(ttp,B1),1),ttp);
B2=4*pi*R22*gv(u,v+c2).*c2+4*pi*GN21*gu(u,v+c2).*c1;
B2p=ppval(fnder(spline(ttp,B2),1),ttp);
uf=@(T) spline(ttp,u,mod(T,period));
upf=@(T) spline(ttp,up,mod(T,period));
vf=@(T) spline(ttp,v,mod(T,period));
vpf=@(T) spline(ttp,vp,mod(T,period));
c1f=@(T) spline(ttp,c1,mod(T,period));
c1pf=@(T) spline(ttp,c1p,mod(T,period));
c1ppf=@(T) spline(ttp,c1pp,mod(T,period));
c2f=@(T) spline(ttp,c2,mod(T,period));
c2pf=@(T) spline(ttp,c2p,mod(T,period));
c2ppf=@(T) spline(ttp,c2pp,mod(T,period));
B1f=@(T) spline(ttp,B1,mod(T,period));
B2f=@(T) spline(ttp,B2,mod(T,period));
B1pf=@(T) spline(ttp,B1p,mod(T,period));
B2pf=@(T) spline(ttp,B2p,mod(T,period));
x1f=@(T) spline(ttp,x1,mod(T,period));
x2f=@(T) spline(ttp,x2,mod(T,period));
x3f=@(T) spline(ttp,x3,mod(T,period));
x4f=@(T) spline(ttp,x4,mod(T,period));
top=@(T) B1f(T-delay).*(x1f(T)+x3f(T))+B2f(T-delay).*(x2f(T)+x4f(T));
bottom=@(T) upf(T).*x1f(T)+vpf(T).*x2f(T)+delay/omega1*(c1pf(T).*(x1f(T)+x3f(T))+c2pf(T).*(x2f(T)+x4f(T)));
omega2=quad(top,0,period)/quad(bottom,0,period);
```

A.6 Comsol Code for Receptor Model in Chapter 4

This code can solve models that involve the patches on the surface of the sphere as well as the sub cellular compartments. This code was used to solve the model in §4.4. Simpler variants of it were used to solve the other models in Chapter 4.

```
function out = model(patchnumb,kk,tout)

import com.comsol.model.*
import com.comsol.model.util.*
model = ModelUtil.create('Model');
model.modelPath('/misc/temp-1/clevy/phd from home directory/Comsol4Cell');
model.modelNode.create('comp1');
```

```
model.geom.create('geom1', 3);
model.mesh.create('mesh1', 'geom1');
model.physics.create('c', 'CoefficientFormPDE', 'geom1', {'u1','u2','u3','u4'});
model.study.create('std1');
model.study('std1').feature.create('time', 'Transient');
model.study('std1').feature('time').activate('c', true);
%Patches Source Function
model.func.create('rect1', 'Rectangle');
model.func('rect1').model('comp1');
model.func('rect1').set('lower', '0');
model.func('rect1').set('upper', '40');
numstatevar=4; %The number of state variables (see above)
Npatches=patchnumb;  %The number of patches
percent=.05;  %100P is the percent of surface area of unit sphere that patches cover
%Load in the rotation angles for sphere
load(['/misc/temp-1/clevy/phd from home directory/Comsol4Cell/Finding RotAng/RotAng_' num2str(100*percent) '-percent']);
patchangle=acos((Npatches-2*percent)/Npatches);  %The area of each patch is 2*pi*(1-cos(patchangle));
zaxisrot=rotationangles(Npatches,1); %Rotate the sphere through the z-axis by this angle.
xaxisrot=rotationangles(Npatches,2);
yaxisrot=rotationangles(Npatches,3);
trange=['range(0,1,' num2str(tout) ')'];
rtol=0.001;
patchflux='(1+(125*u1/(u1+1)))*1/(u4^3+1)';
epsilon=0.1;
K=[6,1,1,1]*kk;
D=[1,1,1,1];
T=[1,1,1,1];
inits=[0,0,0,0];
%Radii and location of compartment/spheres
r=[.1,.1,.1];
x=[0,0,0];
y=[0,0,0];
z=[0.7,0,-0.7];
%Enzyme Kinetc Functions for internal compartments
%enzyfuncs={'1/epsilon*u1/(u1+1)', '1/epsilon*u2/(u2+1)', '1/epsilon*u3/(u3+1)'};
enzyfuncs={'100*u1^2/(u1^2+1)','100*u2^2/(u2^2+1)','100*u3^2/(u3^2+1)'};
%Location of patches are specified in spherical coordinates.
%theta is measured in the xy plane: counter clockwise from the positive
%x-axis. phi is measured from the positive z-axis down into the positive
%side of the xy plane. Get patches uniformly distributed (Thomsons Problem:)
%[V,Tri,~,Ue]=ParticleSampleSphere('N',Npatches);
%This loads in vector V with patch locations
load(['/misc/temp-1/clevy/phd from home directory/Comsol4Cell/Location of Patches/' num2str(Npatches) 'xyz.mat']);
%V holds the location of the patches in three columns xyz
%Convert to spherical
[theta,phi,junk] = cart2sph(V(:,1), V(:,2), V(:,3));
%Matlabs convention measures phi from xy plane toward positive z-axis.
%Therefore we have to tranform this: phi= Pi/2-phi
phi=pi/2-phi;
%Now convert to degrees which is what Comsol is using for patch location
theta = radtodeg(theta);
phi=radtodeg(phi);
%Move some of the patches slightly (only in some cases)
%so that the finial geometry will work.
if and(Npatches==9, percent==.05)
     theta(2)=theta(2); phi(2)=phi(2)+2;
 theta(5)=theta(5)+3; phi(5)=phi(5)-3;
end
if and(Npatches==10, percent==.05)
     theta(2)=theta(2)+3; phi(2)=phi(2)+1.5;
end
if and(Npatches==11, percent==.05)
     phi(2)=phi(2)+.5;
theta(4)=theta(4)+2.5;
theta(8)=theta(8)-.5;
theta(10)=theta(10)-4;
```

```
end
%Size of Patches: The Surface Area of each patch is of the form
%A=2*pi*(1-cos(angle)). If angle=pi then one patch covers the entire sphere
%See notes for what the angle is. 0<angle<pi.
%We will make all patches the same size
angles=linspace(patchangle,patchangle,length(theta));
%Set up the constants in this Model
model.variable.create('var1');
model.variable('var1').model('comp1');
model.variable('var1').set('epsilon', epsilon);
for i=1:length(D);
model.variable('var1').set(['k' num2str(i)], K(i));
model.variable('var1').set(['d' num2str(i)], D(i));
model.variable('var1').set(['t' num2str(i)], T(i));
model.variable('var1').set(['u' num2str(i) 'init'], inits(i));
end
%Create the Shpere/Cell
model.geom('geom1').feature.create('sph1', 'Sphere');
model.geom('geom1').feature('sph1').set('r', '1');
model.geom('geom1').feature('sph1').set('createselection', 'on');
model.geom('geom1').run('sph1');
%We may need to rotate the sphere so that its edges do not %intersect with the edges of the patches.
%This is an issue with Comsoland can only be solved if we purchase the import cad module. for now we
%will rotate the sphere. For now we will just stick with rotations through the z-axis, x-axis, y-axis.
%After this process the Unit Sphere is referred to as rotsph3 because it has gone through 3 rotations.
%Rotate through z axis
model.geom('geom1').feature.create('rotsph1', 'Rotate');
model.geom('geom1').feature('rotsph1').selection('input').set({'sph1'});
model.geom('geom1').feature('rotsph1').set('axistype', 'z');
model.geom('geom1').feature('rotsph1').set('createselection', 'on');
model.geom('geom1').feature('rotsph1').set('rot', zaxisrot);
model.geom('geom1').run('rotsph1');
%Rotate through x axis
model.geom('geom1').feature.create('rotsph2', 'Rotate');
model.geom('geom1').feature('rotsph2').selection('input').set({'rotsph1'});
model.geom('geom1').feature('rotsph2').set('axistype', 'x');
model.geom('geom1').feature('rotsph2').set('createselection', 'on');
model.geom('geom1').feature('rotsph2').set('rot', xaxisrot);
model.geom('geom1').run('rotsph2');
%Rotate through y axis
model.geom('geom1').feature.create('rotsph3', 'Rotate');
model.geom('geom1').feature('rotsph3').selection('input').set({'rotsph2'});
model.geom('geom1').feature('rotsph3').set('axistype', 'y');
model.geom('geom1').feature('rotsph3').set('createselection', 'on');
model.geom('geom1').feature('rotsph3').set('rot', yaxisrot);
model.geom('geom1').run('rotsph3');
%This Loop Creates Patches
for i=1:1:length(theta)
model.geom('geom1').feature.create(['wp' num2str(i)], 'WorkPlane');
model.geom('geom1').feature(['wp' num2str(i)]).set('unite', true);
model.geom('geom1').feature(['wp' num2str(i)]).set('quickplane', 'xz');
model.geom('geom1').feature(['wp' num2str(i)]).set('repairtol', '1.0E-6');
model.geom('geom1').feature(['wp' num2str(i)]).geom.feature.create('pc1', 'ParametricCurve');
model.geom('geom1').feature(['wp' num2str(i)]).geom.feature('pc1').set('parmin', pi/2-angles(i));
model.geom('geom1').feature(['wp' num2str(i)]).geom.feature('pc1').set('parmax', pi/2);
model.geom('geom1').feature(['wp' num2str(i)]).geom.feature('pc1').setIndex('coord', 'cos(s)', 0);
model.geom('geom1').feature(['wp' num2str(i)]).geom.feature('pc1').setIndex('coord', 'sin(s)', 1);
model.geom('geom1').feature(['wp' num2str(i)]).geom.feature('pc1').set('rtol', '1.0E-6'); %Added this in to fix issue
model.geom('geom1').feature(['wp' num2str(i)]).geom.feature('pc1').set('maxknots', '1000');  %Added this in to fix issue
model.geom('geom1').feature(['wp' num2str(i)]).geom.run('pc1');
model.geom('geom1').run(['wp' num2str(i)]);
model.geom('geom1').feature.create(['rev' num2str(i)], 'Revolve');
model.geom('geom1').feature(['rev' num2str(i)]).set('axistype', '3d');
model.geom('geom1').feature(['rev' num2str(i)]).setIndex('axis3', '0', 1);
model.geom('geom1').feature(['rev' num2str(i)]).setIndex('axis3', '1', 2);
model.geom('geom1').feature(['rev' num2str(i)]).set('polres', '50');
```

```
model.geom('geom1').feature(['rev' num2str(i)]).set('createselection', 'on');
model.geom('geom1').run(['rev' num2str(i)]);
model.geom('geom1').feature.create(['rot' num2str(2*i-1)], 'Rotate');
model.geom('geom1').feature(['rot' num2str(2*i-1)]).selection('input').named(['rev' num2str(i)]);
model.geom('geom1').feature(['rot' num2str(2*i-1)]).set('rot', phi(i));
model.geom('geom1').feature(['rot' num2str(2*i-1)]).set('createselection', 'on');
model.geom('geom1').feature(['rot' num2str(2*i-1)]).set('axistype', 'y');
model.geom('geom1').run(['rot' num2str(2*i-1)]);
model.geom('geom1').feature.create(['rot' num2str(2*i)], 'Rotate');
model.geom('geom1').feature(['rot' num2str(2*i)]).selection('input').named(['rot' num2str(2*i-1)]);
model.geom('geom1').feature(['rot' num2str(2*i)]).set('rot', theta(i));
model.geom('geom1').feature(['rot' num2str(2*i)]).set('createselection', 'on');
model.geom('geom1').run(['rot' num2str(2*i)]);
end
%This loop creates sphere compartments inside domain
for i=1:length(r)
model.geom('geom1').feature.create(['sph' num2str(i+1)], 'Sphere');
model.geom('geom1').feature(['sph' num2str(i+1)]).set('r', r(i));
model.geom('geom1').feature(['sph' num2str(i+1)]).setIndex('pos', x(i), 0);  % x location of sphere   center
model.geom('geom1').feature(['sph' num2str(i+1)]).setIndex('pos', y(i), 1);   % y location of sphere  center
model.geom('geom1').feature(['sph' num2str(i+1)]).setIndex('pos', z(i), 2);   % z location of sphere center
model.geom('geom1').feature(['sph' num2str(i+1)]).set('createselection', 'on');
model.geom('geom1').run(['sph' num2str(i+1)]);
end
%Create a cell array of strings corresponding to the patches on the sphere.
for i=1:length(theta)
    patches{i}=['rot' num2str(2*i)];
end
%Create a cell array of strings by adding 'rotsph3' (the unit sphere) as
%the first entry followed by patches
%(defined above)
cellunionpatches=['rotsph3' patches];
%Create a cell array of strings corresponding to the interior
%compartments/spheres within the sphere.
for i=1:length(r)
    compartments{i}=['sph' num2str(i+1)];
end
%Union the patches and create a selection. This is so we can use one flux
%boundary condition later defined on this selection (union of all patches)
model.geom('geom1').feature.create('uni1', 'Union');
model.geom('geom1').feature('uni1').selection('input').set(patches);
model.geom('geom1').feature('uni1').set('createselection', 'on');
model.geom('geom1').run('uni1');
%Union the Patches with the Cell/Sphere. This is used for the Difference
%operator that follows right below this
model.geom('geom1').feature.create('uni2', 'Union');
model.geom('geom1').feature('uni2').selection('input').set( cellunionpatches);  %try 'uni1' for .set()
model.geom('geom1').feature('uni2').set('createselection', 'on');
model.geom('geom1').run('uni2');
if length(r) >= 1
%Take the difference of (Sphere + Patches) with (Interior Compartments)
model.geom('geom1').feature.create('dif1', 'Difference');
model.geom('geom1').feature('dif1').selection('input').set({'uni2'});
model.geom('geom1').feature('dif1').selection('input2').set(compartments);
model.geom('geom1').feature('dif1').set('createselection', 'on');
model.geom('geom1').run('dif1');
end
%mphgeom(model, 'geom1','Facealpha',.4,'build','off');
%Form union of geometry
  model.geom('geom1').feature('fin').set('repairtol', '1.0E-6');
  model.geom('geom1').run('fin');
%Create the Boundary Condition on the patches. The patches were formed
%into uni1 (union 1)
model.physics('c').feature.create('flux1', 'FluxBoundary', 2);
model.physics('c').feature('flux1').selection.named('geom1_uni1_bnd');
model.physics('c').feature('flux1').set('g', 1, patchflux);
```

```
%Boundary conditions using enzyme functions on compartments
for i=1:length(r)
model.physics('c').feature.create(['flux' num2str(i+1)], 'FluxBoundary', 2);
model.physics('c').feature(['flux' num2str(i+1)]).selection.named( ['geom1_sph' num2str(i+1) '_bnd']);
model.physics('c').feature(['flux' num2str(i+1)]).set( 'g', i+1, enzyfuncs(i));
en
%Set up the PDEs and coefficients
for i=1:length(D);
%Turn off source term
model.physics('c').feature('cfeq1').set('f', i, '0');
%Set up Decay terms
model.physics('c').feature('cfeq1').set('a', (length(D)+1)*i-(length(D)), ['k' num2str(i)]);
%Set up Term in front of du/dt
model.physics('c').feature('cfeq1').set('da', (length(D)+1)*i-(length(D)), ['t' num2str(i)]);
%Set up initial conditions
model.physics('c').feature('init1').set(['u' num2str(i)], 1, ['u' num2str(i) 'init']);
% %Set up diffusion terms
 model.physics('c').feature('cfeq1').set('c', (length(D)+1)*i-(length(D)), ...
 {['d' num2str(i)] '0' '0' '0' ['d' num2str(i)] '0' '0' '0' ['d' num2str(i)]});
end
%This line turns on the progress log
ModelUtil.showProgress(true);
%This loop creates the probe plots so we can view the solutions while
%solving
for i=1:numstatevar
model.probe.create(['pdom' num2str(i)], 'DomainPoint');
model.probe(['pdom' num2str(i)]).model('comp1');
model.probe(['pdom' num2str(i)]).setIndex('coords3', -.35, 0, 0);
model.probe(['pdom' num2str(i)]).setIndex('coords3', -.65, 0, 1);
model.probe(['pdom' num2str(i)]).setIndex('coords3',-.65, 0, 2);
model.probe(['pdom' num2str(i)]).feature(['ppb' num2str(i)]).set('expr', ['u' num2str(i)]);
model.probe(['pdom' num2str(i)]).genResult([]);
model.result.numerical(['pev' num2str(i)]).set('table', 'tbl1');
model.result.numerical(['pev' num2str(i)]).set('innerinput', 'all');
model.result.numerical(['pev' num2str(i)]).set('outerinput', 'all');
end
%Create a cell array of strings corresponding to the
%different probe plots (created above)
%for each state variable.
for i=1:length(numstatevar)
    probeplots{i}=['pdom' num2str(i)];
end
%Set up the Study/Solvers
model.study('std1').feature('time').set('tlist', trange);
model.study('std1').feature('time').set('rtolactive', 'on');
model.study('std1').feature('time').set('rtol', rtol);
model.study('std1').feature('time').set('plot', 'on');
model.study('std1').feature('time').set('plotfreq', 'tsteps');
model.result('pg1').set('window', 'window1');
model.result('pg1').feature('tblp1').set('legend', 'on');
model.result('pg1').run;
model.sol.create('sol1');
model.sol('sol1').study('std1');
model.sol('sol1').feature.create('st1', 'StudyStep');
model.sol('sol1').feature('st1').set('study', 'std1');
model.sol('sol1').feature('st1').set('studystep', 'time');
model.sol('sol1').feature.create('v1', 'Variables');
model.sol('sol1').feature('v1').set('control', 'time');
model.shape('shape1').feature('shfun1');
model.sol('sol1').feature.create('t1', 'Time');
model.sol('sol1').feature('t1').set('tlist', trange);
model.sol('sol1').feature('t1').set('plot', 'on');
model.sol('sol1').feature('t1').set('plotgroup', 'pg1');
model.sol('sol1').feature('t1').set('plotfreq', 'tsteps');
model.sol('sol1').feature('t1').set('probesel', 'all');
model.sol('sol1').feature('t1').set('probes', probeplots);
```

```
model.sol('sol1').feature('t1').set('probefreq', 'tsteps');
model.sol('sol1').feature('t1').set('control', 'time');
model.sol('sol1').feature('t1').feature.create('fc1', 'FullyCoupled');
model.sol('sol1').feature('t1').feature('fc1').set('linsolver', 'dDef');
model.sol('sol1').feature('t1').feature.remove('fcDef');
model.sol('sol1').attach('std1');
model.sol('sol1').runAll;
 out = model;
```