A Homotopy-Minimization Method for Parameter Estimation in Differential Equations and Its Application in Unraveling the Reaction Mechanism of the Min System

by

William Christopher Carlquist

BS in Biology, The University of Utah, 2008
BS in Mathematics, The University of Utah, 2008
M.Sc. in Mathematics, The University of British Columbia, 2012

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

The Faculty of Graduate and Postdoctoral Studies

(Mathematics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

January 2019

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the dissertation entitled:

A Homotopy-Minimization Method for Parameter Estimation in Differential Equations and Its Application in Unraveling the Reaction Mechanism of the Min System

submitted by William Christopher Carlquist in partial fulfillment of the requirements for the degree of Doctor of Philosophy

in Mathematics

Examining Committee:

Eric Cytrynbaum, Mathematics
Supervisor

Leah Keshet, Mathematics
Supervisory Committee Member

Colin Macdonald, Mathematics
University Examiner

Carl Michal, Physics and Astronomy
University Examiner

David Campbell, Statistics and Actuarial Science at Simon Fraser University
External Examiner

Additional Supervisory Committee Members:

Daniel Coombs, Mathematics
Supervisory Committee Member

Abstract

A mathematical model of a dynamical process, often in the form of a system of differential equations, serves to elucidate underlying dynamical structure and behavior of the process that may otherwise remain opaque. However, model parameters are often unknown and may need to be estimated from data for a model to be informative. Numerical-integration-based methods, which estimate parameters in a differential equation model by fitting numerical solutions to data, can demand extensive computation, especially for large stiff systems that require implicit methods for stability. Non-numerical integration methods, which estimate parameters in a differential equation model by fitting solution approximations to data, do not provide an impartial measure of how well a model fits data, a measure required for the testability of a model. In this dissertation, I propose a new method that steps back from a numerical-integration-based method, and instead allows an optimal data-fitting numerical solution to emerge as part of an optimization process. This method bypasses the need for implicit solution methods, which can be computationally intensive, seems to be more robust than numerical-integration-based methods, and, interestingly, admits conservation principles and integral representations, which allow me to gauge the accuracy of my optimization.

The *Escherichia coli* Min system is one of the simplest known biological systems that demonstrates diverse complex dynamic behavior or transduces local interactions into a global signal. Various mathematical models of the Min system show behaviors that are qualitatively similar to dynamic behaviors of the Min system that have been observed in experiments, but no model has been quantitatively compared to time-course data. In this dissertation, I extract time-course data for model fitting from experimental measurements of the Min system and fit established and novel biochemistry-based models to the time-course data using my parameter estimation method for differential equations. Comparing models to time-course data allows me to make precise distinctions between biochemical assumptions in the various models. My modeling and fitting supports a novel model, which suggests that a regular, ordered, stability-switching mechanism underlies the emergent, dynamic behavior of the Min system.

Lay Summary

In this dissertation, I develop a method to map experimental measurements onto mathematical models that describe how the measured system changes in time and space. This mapping allows me to test whether a mathematical model can explain experimental observations and helps understand the underlying dynamic structure of a modeled system. After developing and testing my method, I apply it to map experimental measurements of a protein system that demonstrates interesting dynamic behavior onto established and novel mathematical models that describe the protein systems's temporal evolution. My modeling and data mapping inform a novel mechanism that may underlie the dynamic behavior of the protein system.

Preface

William Carlquist performed the research in this dissertation, designed and wrote the computer programs for this dissertation, and wrote this dissertation. Eric Cytrynbaum provided research discussion and feedback on the writing. Vassili Ivanov and Kiyoshi Mizuuchi provided the previously published experimental data used in Chapter 4, Appendix F, and Appendix G. The research in this dissertation is original and unpublished.

Table of Contents

Appendices

List of Tables

List of Figures

Acknowledgements

My research supervisor, Eric Cytrynbaum, introduced me to the Min system and collocation methods. Throughout my research, he gave me the freedom to explore while providing meaningful discussion to keep me from getting lost. I sincerely thank him. My supervisory committee members, Leah Edelstein-Keshet and Daniel Coombs, encouraged me, provided helpful discussion, and fostered an enriching MathBio group at UBC. I thank them. Vassili Ivanov and Kiyoshi Mizuuchi kindly provided me with their experimental data, and I thank them. Without the love and support from my family and my better half, Şule, I would not have started or completed my PhD program. I cannot thank them enough.

for Şule

Chapter 1

Introduction

A mathematical model of a dynamical process serves to elucidate underlying dynamical structure and behavior of the process that may otherwise remain opaque. Across different fields, notably in Biology, systems of interest are becoming more interrelated, with a commensurate increase in mathematical model complexity. Often, an explicit model for the evolution of a complex dynamical process may not be known or may not exist. Alternatively, from experiment or postulate, one may formulate an implicit model for the evolution of a complex dynamical process, relating the state of the system to the change in the state of the system. Such models fall into one of a variety of forms including difference equations, stochastic processes, and, most ubiquitously, differential equations. A differential equation model of a complex dynamical process, with a large number of states, parameters, and nonlinearities, often admits solutions that are sensitive to changes in parameter values and often contains parameters that are not directly measurable by experiment. As such, parameter values in a differential equation model of a complex dynamical process are often unknown, and dynamical outcomes of the model are not well characterized.

Fitting solutions of a differential equation model to data determines parameter values for a model. For a good model, fitted parameter values will confer dynamical structure on the model so that the model fits data well. In converse, the ability of a model to fit data provides testability for a model. In this dissertation, I develop a parameter estimation method for differential equations that accurately estimates model parameter values from data and accurately estimates a model's ability to fit data.

The *Escherichia coli* Min system is one of the simplest known biological systems that demonstrates diverse complex dynamic behavior or transduces local interactions into a global signal. As such, the Min system is currently one of the most reduced model systems for understanding such behaviors. I apply my parameter estimation method for differential equations to fit established and novel models of the Min system to time-course data. My modeling and fitting reveals a novel mechanism that may underlie the dynamic behavior of the Min system.

1.1 Parameter Estimation in Differential Equations

Various parameter estimation methods exist for differential equations. Most commonly, numerical solutions of differential equations are fit to data, as, generally, closed-form solutions to differential equations are not known or do not exist for fitting. In numerical-integration-based

methods, parameter values are iteratively updated to minimize a measure of the difference between numerical solution values and data [11]. Numerical-integration-based methods are precise, in that numerical solution values are directly compared to data, so parameter estimates correspond directly to the numerical solution that fits data best. However, complex systems of differential equations admit a variety of parameter-dependent solution behaviors, and bifurcations separate the numerical solution space into regions with qualitatively different behaviors. Thus, to find parameter values of the optimal data-fitting numerical solution, initial parameter-value estimates must be chosen for a numerical solution with the same qualitative behavior as the optimal data-fitting numerical solution, as local search information is lost across bifurcations. As such, numerical-integration-based methods require extensive parameter search-space probing, which, in practice, is accomplished by combining global optimization methods, such as genetic algorithms, with local numerical-integration-based methods [49] [78]. However, repeated numerical integration is computationally intensive, especially for large stiff systems of differential equations that require implicit methods for stability, and excessively long computational times in numerical-integration-based methods may surpass tractability.

Non-numerical integration methods for differential equation parameter estimation, such as collocation methods, relax the exactness of using numerical solutions to increase reliability in parameter estimation and to gain computational efficiency. Static collocation methods are the computationally simplest form of collocation method and are used to estimate parameters in systems of differential equations from data with measurements of all model states. In them, to generate smooth solution proxies, focusing on fitting non-local data behavior, smooth splines with fewer knots than data points are fit to data [71], or focusing on fitting local data behavior, polynomials centered at data points are fit to data [43]. Then, parameter values in differential equations are estimated by minimizing a measure of satisfying the system of differential equations with the solution proxies as state values.

Dynamic collocation methods extend the idea of static collocation methods to estimate parameters in systems of differential equations with unobserved states by incorporating a dynamic basis representation for each state, a linear combination of basis functions, generally splines. In them, under some smoothing penalty, basis representations of states are fit to data under a fixed set of parameter values. Then, parameter values are re-estimated, either by minimizing a measure of satisfying the system of differential equations with the basis representations as state values [57] or by fitting basis representations of states, treated as implicit functions of model parameters, to data [58], and the process is repeated. Generally, smoothing penalties in data fitting consist of a weighted measure of satisfying the system of differential equations with the basis representations as state values, where a small penalty weight biases fitting towards data and a large penalty weight biases fitting towards a solution of the system of differential equations. For reliable parameter estimation, the penalty weight is chosen relatively large. However, an excessively large penalty weight obscures data fitting. Parameter estimates directly depend on the penalty weight, so the penalty weight is chosen judiciously. Often, the penalty weight

is incrementally increased until some stopping criterion is met: when basis representations of states begin to deform after stabilizing [58], parameter estimates become stable [77] then begin to destabilize [9], or when a sharp decrease in data fitting accommodates a sharp increase in satisfying the system of differential equations [7]. The penalty weight may also be chosen as the penalty weight that minimizes data fitting error under some cross-validation criterion, such as model based complexity [10] or error in satisfying the system of differential equations [79]. Alternatively, with a Bayesian approach, the posterior conditional probability density, given the data, may be generated by assuming some prior probability distribution for the penalty weight in addition to prior probability distributions for model parameters [79]. Or, multiple posterior conditional probability densities may be simultaneously generated under different penalty weights, with exchange, to more robustly estimate the posterior conditional probability density for some large penalty weight [8].

The choice of method for parameter estimation in differential equations depends on the data, the model, and the motivation for parameter estimation. If parameter estimation is motivated by measuring the underlying parameter values of some dynamic process, then a collocation method will often return reliable parameter estimates. However, non-numerical integration methods, such as collocation methods, approximate parameter values of the optimal data-fitting solution through an approximation of the optimal data-fitting solution. So, even though parameter estimates may be similar to those of the optimal data-fitting numerical solution, parameter estimates from non-numerical integration methods may admit numerical solutions with significantly different behavior than the optimal data-fitting numerical solution, especially in complex systems with sensitivity to parameters. Also, because non-numerical integration methods approximate the optimal data-fitting solution, they do not assess how well the optimal data-fitting numerical solution fits data. To reliably compare different models of the same dynamic process, each model's ability to fit the data must be assessed, requiring the determination of the optimal data-fitting numerical solution. However, as mentioned previously, calculating the optimal data-fitting numerical solution using a numerical-integration-based method can require an excessively long computational time, especially with a large system of differential equations that requires an implicit method for stability.

In this dissertation, I propose a method that extends the idea of collocation methods to allow me to calculate the optimal data-fitting numerical solution and its parameters for a differential equation model. It steps back from a numerical-integration-based method, and instead allows the numerical solution to emerge as part of an optimization process. This method bypasses the need to calculate numerical solutions with implicit methods, which can be computationally intensive, seems to be more robust than numerical-integration-based methods, and, interestingly, admits conservation principles and integral representations, which allow me to gauge the accuracy of my optimization.

1.2 The Min System

The Min system, consisting of three proteins, MinC, MinD, and MinE, dynamically orients the site of cell division toward midcell in *Escherichia coli*. Local interactions of MinD and MinE on the cell membrane drive a recurring, coordinated repositioning of MinD and MinE from cell pole to cell pole [60]. MinC disrupts the aggregation of FtsZ into the Z-ring ([14], [3], [33], [30], [55]), the contractile ring that divides the cell, and localizes to the cell membrane in the presence of MinD ([35], [29], [30], [34], [42]). The pole-to-pole repositioning of MinD shuttles MinC from cell pole to cell pole [29]. Over time, the average concentration of MinC is higher at cell poles than at midcell, leading to greater inhibition of Z-ring formation at cell poles than at midcell and dynamically orienting the site of cell division toward midcell ([80], [55]). Ultimately, cell division at midcell produces viable, symmetric daughter cells. A schematic diagram of division-site regulation by the Min system is shown in Figure 1.1.

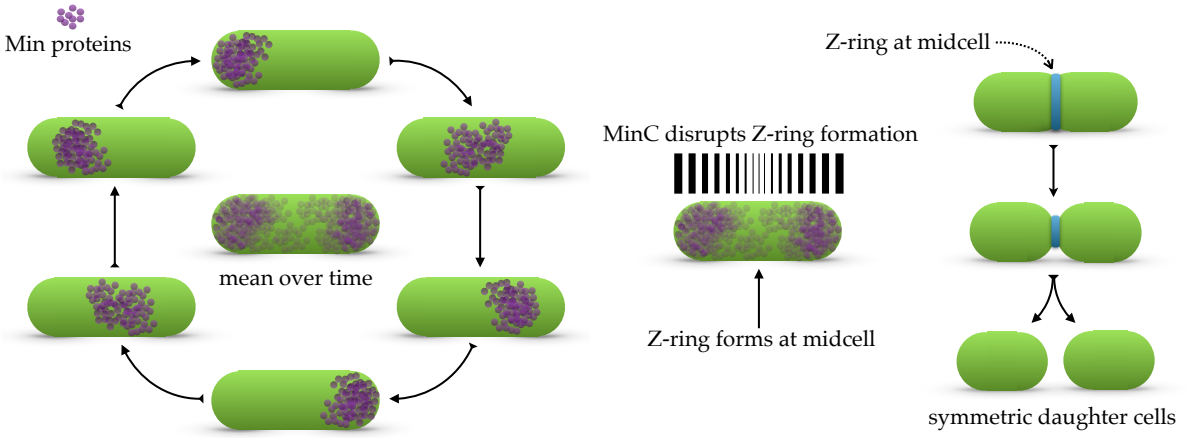


Figure 1.1: Division-site regulation by the Min system. Min proteins oscillate from cell pole to cell pole, with minimum concentrations over time at midcell (left). MinC inhibits the formation of the Z-ring, causing the Z-ring to form at midcell (middle). The Z-ring at midcell contracts to divide the cell into two symmetric daughter cells (right).

The Min system demonstrates interesting dynamic behavior *in vivo* and *in vitro*. In short cells, MinD and MinE arrange into dynamic protein bands that stochastically switch together from cell pole to cell pole [21]. As cells grow longer, stochastic pole-to-pole switching of MinD and MinE stabilizes into regular pole-to-pole oscillations of MinD and MinE ([60], [23], [21]). In cells devoid of FtsZ, cells continually grow, and regular pole-to-pole oscillations of MinD and MinE form into stable pole-to-midcell oscillations of MinD and MinE ([60], [23]). However, the oscillatory behavior of MinD and MinE in cells can be altered by changing the expression levels of MinD and MinE. At low expression levels, MinD and MinE oscillate regularly from cell pole to cell pole in short cells and from cell pole to midcell in long cells; at high induction levels, MinD and MinE stochastically switch from cell pole to cell pole in short cells and from cell pole

to midcell in long cells [69]. Generally, *Escherichia coli* cells are rod shaped. In round mutant cells, MinD and MinE oscillate antipodally ([12],[68]), and in branched mutant cells, MinD and MinE oscillate from branch to branch to branch [72]. On supported lipid bilayers *in vitro*, MinD and MinE arrange into dynamic protein aggregates that oscillate [38] or form into traveling waves ([45], [38], [44], [74], [73]), spiral waves ([45], [38], [44], [74], [73]), dynamic amoeba-like shapes ([38], [73]), snake-like projections [38], mushroom-like shapes [73], and bursts [73]. The oscillatory behavior of MinD and MinE in cells has been reproduced in artificial, rod-shaped, membrane-clad compartments with dimensions on scales that are ten times longer than those in living cells. In compartments with smaller aspect ratios, MinD and MinE oscillate from compartment end to compartment end; in compartments with larger aspect ratios, MinD and MinE oscillate from compartment end to compartment middle [82].

Both *in vivo* and *in vitro*, MinD and MinE form dynamic protein arrangements on spatial scales that are thousands of times larger than the spatial scale of an individual MinD or MinE protein, which are sustained on temporal scales that are much longer than the temporal scale of an individual MinD or MinE interaction on the membrane. Biochemical experiments have elucidated the functional role of the Min system and much of its underlying biochemical basis. However, biochemical experiments only show small-scale snapshots of the reactions that drive dynamic behavior on much larger spatial and temporal scales – crystal structures show stable, static protein configurations and mutational analyses measure amino acid function with respect to a particular functional assay. Protein visualizations, on the other hand, allow for the observation of dynamic behavior, the collective outcome from local reactions, but provide little insight into the local reactions themselves. As such, the direct connection between local reactions and global, dynamic behavior in the Min system is unclear. Mathematical models can predict dynamic outcomes for a set of reactions, and thus provide a means to connect information about local reactions to global, dynamic behavior in the Min system.

Various mathematical models demonstrate dynamic behavior that is qualitatively similar to experimental observations of the Min system. Most mathematical models have focused on behavior of the Min system *in vivo*. On domains approximating short rod-shaped cells, agent-based models demonstrate MinD and MinE densities that stochastically switch together from domain pole to domain pole ([21], [4]). As short rod-shaped domains grow, MinD and MinE densities transition from static to regular pole-to-pole oscillations in deterministic models [75] that are sustained in both deterministic and stochastic models on mid-sized rod-shaped domains ([26], [48], [41], [64]). As mid-sized rod-shaped domains grow, regular pole-to-pole oscillations of MinD and MinE densities transition into regular pole-to-mid-domain oscillations in deterministic models ([48], [4], [75]) that are sustained in both deterministic and stochastic models on long rod-shaped domains ([48], [36], [47], [70]). Additionally, deterministic and stochastic models have qualitatively addressed various other aspects of *in vivo* oscillatory behavior in the Min system: oscillatory behavior in round mutant cells ([37], [20], [22], [4]), oscillatory behavior in branched mutant cells [72], oscillatory behavior in flattened, irregular cells [62], oscillatory

behavior in dividing cells ([70], [65], [17], [75]), oscillatory behavior of MinE mutants ([13], [1]), transitions in oscillation waveforms [75], midcell establishment through oscillation ([26], [25], [40], [22]), and the dependence of oscillation period on protein numbers ([26], [36], [70], [40]), cell length ([26], [70]), and temperature ([22], [75]). Several mathematical models have focused on behavior of the Min system *in vitro*. On domains approximating supported lipid bilayers, MinD and MinE densities form into traveling waves ([45], [54]) and spiral waves ([45], [4]) in deterministic models. Additionally, deterministic and stochastic models have qualitatively addressed MinD and MinE patterning *in vitro* on geometrically confined membranes [63] and on micropatterned substrates [24].

Most mathematical models of the Min system are based on the biochemistry-based characterization that MinE acts as an inhibitor of MinD membrane binding: cytosolic MinD monomers bind to ATP and form dimers ([34], [81]), which bind to the membrane ([28], [34], [32], [42], [81], [45]); MinE dimers bind to MinD dimers on the membrane ([28], [44]) and stimulate ATPase activity in MinD dimers, causing MinD dimers to separate and dissociate from the membrane ([31], [28], [34], [42]). MinE acting as an inhibitor of MinD membrane binding is depicted in Figure 1.2. Recent experiments have shown, however, that MinE can act to both stabilize and inhibit MinD membrane binding, with MinE stabilizing MinD membrane binding at lower relative concentrations of MinE to MinD and MinE inhibiting MinD membrane binding at higher relative concentrations of MinE to MinD [73]. No mathematical model has accounted for MinE's dual role in MinD membrane binding and its biological implications remain unknown.

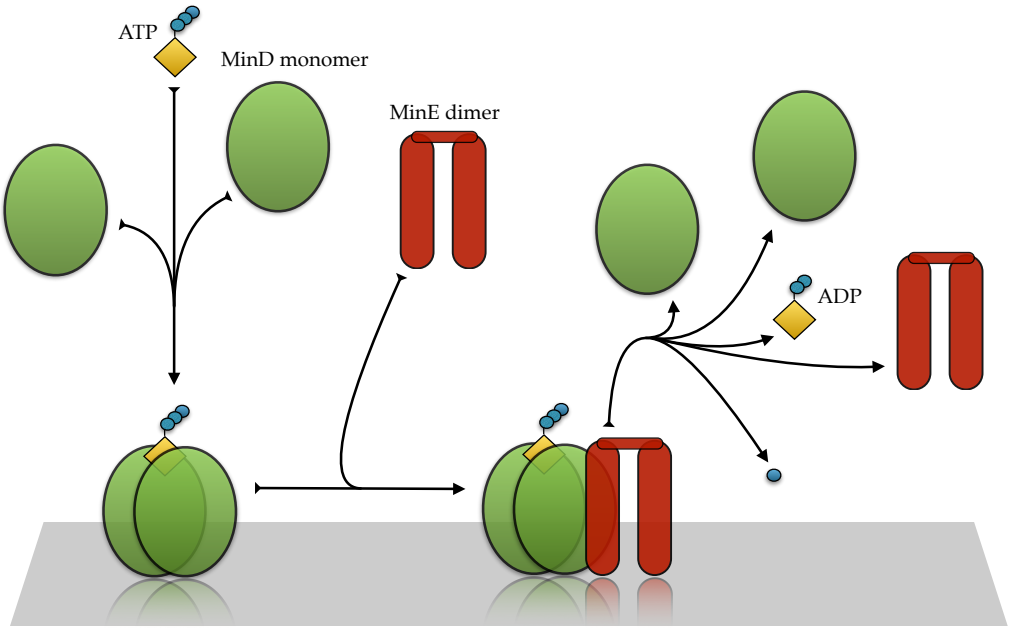


Figure 1.2: MinE acting as an inhibitor of MinD membrane binding. Cytosolic MinD monomers bind to ATP and form dimers, which bind to the membrane (left). MinE dimers bind to MinD dimers on the membrane (center) and stimulate ATPase activity in MinD dimers, causing MinD dimers to separate and dissociate from the membrane (right). This classic model does not account for the dual stabilizing and inhibitory roles of MinE.

Various quantitative experimental measurements have been used to validate mathematical models of the Min system: pole-to-pole oscillation period *in vivo* ([47], [70], [13], [5], [1], [22], [4], [75]), distributions of residence times during stochastic pole-to-pole switching ([21], [4]) and regular pole-to-pole oscillations [21] *in vivo*, and traveling wave velocity and wavelength *in vitro* [45]. However, no model has been quantitatively compared to time-course data. A large variety of models demonstrate dynamic behavior that is qualitatively similar to experimental observations without accounting for observed biological phenomena such as MinE's dual role in MinD membrane binding. A quantitative comparison of models to time-course data is the next logical step in unraveling how proposed reactions contribute to Min system dynamics.

In this dissertation, I extract time-course data for model fitting from Ivanov and Mizuuchi's *in vitro* experimental measurements of the Min system [38]. I fit established and novel biochemistry-based models to the time-course data using my parameter estimation method for differential equations. Comparing models to time-course data allows me to make precise distinctions between biochemical assumptions in the various models. My modeling and fitting supports a novel model that accounts for MinE's previously unmodeled dual role as a stabilizer and an inhibitor of MinD membrane binding. It suggests that a regular, ordered, stability-switching mechanism underlies the emergent, dynamic behavior of the Min system.

1.3 Chapter Summaries

- In Chapter 2, I develop a method that allows me to calculate the optimal data-fitting numerical solution and its parameters for a differential equation model without using numerical integration. Additionally, I show that my method admits conservation principles and integral representations that allow me to gauge the accuracy of my optimization.

- In Chapter 3, I test my method using a system of first order ordinary differential equations, a system of second order ordinary differential equations, and a system of partial differential equations. In doing so, I compare the performance of my method to that of an analogous numerical-integration-based method, explore how my method can inform modeling insufficiencies and potential model improvements, and expound how conservation principles and integral representations in my method gauge the accuracy of my optimization in practice.

- In Chapter 4, I briefly summarize extracting time-course data for model fitting from experimental measurements of the Min system. I fit established and novel biochemistry-based models to the time-course data using my method. In doing so, I explore how individual reactions affect a model's ability to describe the time-course data. Based on my results, I interpret a novel mechanism that may underlie the dynamic behavior of the Min system.

- In Chapter 5, I briefly summarize my results from the previous chapters and discuss limitations and extensions of my method and fitting models of the Min system to time-course data.

Chapter 2

A Homotopy-Minimization Method for Parameter Estimation in Differential Equations

2.1 Introduction

Non-numerical integration methods estimate parameters in a differential equation model by fitting solution approximations to data. Often, non-numerical integration methods, such as collocation methods, will return reliable parameter estimates. However, because non-numerical integration methods approximate the optimal data-fitting solution, they do not provide an impartial measure of how well a differential equation model fits data, a measure required for the testability of a model. Numerical-integration-based methods estimate parameters in a differential equation model by fitting numerical solutions to data. As such, numerical-integration-based methods directly find the optimal data-fitting numerical solution and its parameter for a differential equation model. However, numerical-integration-based methods can demand extensive computation, especially for large stiff systems of differential equations that require implicit methods for stability. In this chapter, I develop a method that allows me to calculate the optimal data-fitting numerical solution and its parameters for a differential equation model without using numerical integration. In doing so, my method bypasses the need to calculate numerical solutions with implicit methods, which can be computationally intensive. Additionally, in this chapter, I show that my method admits conservation principles and integral representations that allow me to gauge the accuracy of my optimization.

2.2 Method Overview

Here, for simplicity in presentation, I explicate the method for a system of first order ordinary differential equations. I extend the method to systems of higher order ordinary differential equations and systems of partial differential equations in Sections A.1 and A.2.

A first order ordinary differential equation model of some dynamic process in t, with n_x states, $x_1, x_2, \ldots, x_{n_x}$, n_p parameters, $p_1, p_2, \ldots, p_{n_p}$, and n_y observable states, $y_1, y_2, \ldots, y_{n_y}$,

is defined by the system of equations,

$$F_i\left(t, p_1, p_2, \ldots, p_{n_p}, x_1, x_2, \ldots, x_{n_x}, \frac{dx_1}{dt}, \frac{dx_2}{dt}, \ldots, \frac{dx_{n_x}}{dt}\right) = 0, \tag{2.1a}$$

$$y_j = g_j\left(p_1, p_2, \ldots, p_{n_p}, x_1, x_2, \ldots, x_{n_x}\right), \tag{2.1b}$$

where functions F_i, for $i \in \{1, 2, \ldots, n_x\}$, provide a model for the evolution of state values, and the functions g_j, for $j \in \{1, 2, \ldots, n_y\}$, define observable states. For some observed data values, $\mathrm{y}_{1,k}, \mathrm{y}_{2,k}, \ldots, \mathrm{y}_{n_y,k}$, measured at times t_k, for $k \in \{1, 2, \ldots, n_t\}$, I seek the parameters $p_1, p_2, \ldots, p_{n_p}$ such that the functions $x_i(t, p_1, p_2, \ldots, p_{n_p})$, for $i \in \{1, 2, \ldots, n_x\}$, satisfy differential equation system (2.1a) and admit the observable state values that most closely approximates the observed data, in some sense.

Generally, solutions to the system of equations (2.1a) are difficult or impossible to find in closed form. However, using a numerical approximation method – finite difference, finite element, etc – I can numerically approximate the value of solutions at time t_k, for $k \in \{1, 2, \ldots, n_t\}$. Because data may be sampled more sparsely than that required for the desired numerical solution accuracy, I choose the numerical discretization $\{t_k : k \in \mathcal{I}_\Delta\}$ to be a refinement of $\{t_k : k \in \{1, 2, \ldots, n_t\}\}$, where the index set of the numerical discretization, $\mathcal{I}_\Delta$, is a super set of $\{1, 2, \ldots, n_t\}$. In doing so, I index grid points in $\{t_k : k \in \mathcal{I}_\Delta\}$ that lie between adjacent grid points in $\{t_k : k \in \{1, 2, \ldots, n_t\}\}$ with fractional indices that reflect their relative location within the discretization. Once a numerical method is chosen, equation (2.1a) can be formulated into a method-dependent system of equations for the discrete numerical solution values $x_{i,k}$:

$$\begin{aligned} f_{i,k}\left(t_1, \ldots, t_{n_t}, p_1, \ldots, p_{n_p}, x_{1,1}, x_{2,1}, \ldots, x_{n_x,1}, x_{1,2}, \ldots, x_{n_x,n_t}\right) = \\ f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) = 0, \end{aligned} \tag{2.2}$$

for all $i \in \{1, 2, \ldots, n_x\}$ and for all $k \in \mathcal{I}_\Delta$.

Once a measure of the quality of the model's fit to the data is chosen – least-squares, negative log-likelihood, etc – I can define a functional $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ with the properties that (i) $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x}) \geq 0$, (ii) $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x}) = 0$ if and only if $g_j(\mathbf{p}, x_{1,k}, \ldots, x_{n_x,k}) = \mathrm{y}_{j,k}$ for all $j \in \{1, 2, \ldots, n_y\}$ and for all $k \in \{1, 2, \ldots, n_t\}$, and (iii) $r_{\mathrm{y}}(\mathbf{p}_1, \mathbf{x}_1) < r_{\mathrm{y}}(\mathbf{p}_2, \mathbf{x}_2)$ implies that $(\mathbf{p}_1, \mathbf{x}_1)$ gives a better fit to the data than does $(\mathbf{p}_2, \mathbf{x}_2)$. I describe the construction of $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ using a normalized least-squares measure in Section 2.3. Ultimately, I seek the parameters $\check{\mathbf{p}}$, which minimize $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ subject to the constraints $f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) = 0$ for all $i \in \{1, 2, \ldots, n_x\}$ and for all $k \in \mathcal{I}_\Delta$, the parameters of the numerical solution that fits the data best.

The structure of differential equation solutions can cause numerical-integration-based methods to be inaccurate or inefficient. High dimensional, nonlinear systems of differential equations with many parameters often contain bifurcations, which separate the solution space, and thus the numerical solution space, into regions with different qualitative behavior. Bifurcations effectively disconnect regions within the numerical solution space, obfuscating optimization

with a numerical-integration-based method. Also, numerical-integration-based methods require repeated numerical integration, which can be computationally very expensive, especially for differential equations that require implicit methods for stability. To ameliorate inaccuracies from bifurcations and inefficiencies from calculating numerical solution values, rather than searching for $\breve{\mathbf{p}}$ within the numerical solution space of differential equation system (2.1a), I search for $\breve{\mathbf{p}}$ within an extended space of discrete state values, including discrete state values outside of the solution space of system (2.2). To do so, using some measure of satisfying the numerical solution to differential equation system (2.1a), I can define a functional $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ with the properties that (i) $r_{\Delta x}(\mathbf{p}, \mathbf{x}) \geq 0$, (ii) $r_{\Delta x}(\mathbf{p}, \mathbf{x}) = 0$ if and only if $f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) = 0$ for all $i \in \{1, 2, \ldots, n_x\}$ and for all $k \in \mathcal{I}_\Delta$, and (iii) $r_{\Delta x}(\mathbf{p}_1, \mathbf{x}_1) < r_{\Delta x}(\mathbf{p}_2, \mathbf{x}_2)$ implies that $(\mathbf{p}_1, \mathbf{x}_1)$ satisfies the numerical solution method better than $(\mathbf{p}_2, \mathbf{x}_2)$ does. I describe the construction of $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ using a normalized least-squares measure in Section 2.4. Then, I combine $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ and $r_{\Delta x}(\mathbf{p}, \mathbf{x})$, with proportionality value $\lambda \in (0, 1)$, into a single functional,

$$\rho(\mathbf{p}, \mathbf{x}; \lambda) = (1 - \lambda) r_{\mathrm{y}}(\mathbf{p}, \mathbf{x}) + \lambda r_{\Delta x}(\mathbf{p}, \mathbf{x}), \tag{2.3}$$

a homotopy between $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ and $r_{\Delta x}(\mathbf{p}, \mathbf{x})$, a continuous deformation from $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ to $r_{\Delta x}(\mathbf{p}, \mathbf{x})$. I note, for consistency in scale and units in $\rho(\mathbf{p}, \mathbf{x}; \lambda)$, that I define $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ and $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ on the same scale with the same units. As elaborated in Section 2.8, as $\lambda \to 1^-$ the parameters and state values that minimize $\rho(\mathbf{p}, \mathbf{x}; \lambda)$ approach the parameters and state values of the optimal data-fitting numerical solution.

For $\lambda = 1$, $\rho(\mathbf{p}, \mathbf{x}; \lambda)$ attains a minimum value of zero at all points where $r_{\Delta x}(\mathbf{p}, \mathbf{x}) = 0$, the infinite set of numerical solutions that correspond to the infinite set of all parameter combinations. Thus, for λ near 1, an iterative method like gradient descent will converge to a minimum that sits near some numerical solution with parameter values close to the initial set of parameter values. As such, an iterative method like gradient descent will not likely converge to the global minimizer of $\rho(\mathbf{p}, \mathbf{x}; \lambda)$ for λ near 1. Instead, I minimize $\rho(\mathbf{p}, \mathbf{x}; \lambda)$ over a broad range of λ values, and use the minimization of $\rho(\mathbf{p}, \mathbf{x}; \lambda)$ with smaller values of λ to direct the minimization of $\rho(\mathbf{p}, \mathbf{x}; \lambda)$ with λ near 1, an idea that is similar in spirit to homotopy methods, which start at a solution of a simpler problem and sequentially step toward the solution of a more difficult problem that is homotopic to the simpler problem [18]. However, rather than minimizing $r(\mathbf{p}, \mathbf{x}; \lambda)$ sequentially over an array of λ values, I simultaneously minimize $r(\mathbf{p}, \mathbf{x}; \lambda)$ over an array of λ values, to avoid error propagation from sequential minimization and to avoid excessive computation in ensuring the global minimum of $r(\mathbf{p}, \mathbf{x}; \lambda)$ for a smaller value of λ before beginning the minimization of $r(\mathbf{p}, \mathbf{x}; \lambda)$ for a larger value of λ. I outline my minimization technique in Section 2.7.

For $\lambda \in (0, 1)$, the parameters and state values that minimize $\rho(\mathbf{p}, \mathbf{x}; \lambda)$, $\breve{\mathbf{p}}^\lambda$ and $\breve{\mathbf{x}}^\lambda$, allow me to define useful functions, $\breve{\rho}(\lambda)$, $\breve{r}_{\mathrm{y}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$, as follows:

$$\breve{\rho}(\lambda) = (1 - \lambda) \breve{r}_{\mathrm{y}}(\lambda) + \lambda \breve{r}_{\Delta x}(\lambda) = (1 - \lambda) r_{\mathrm{y}}(\breve{\mathbf{p}}^\lambda, \breve{\mathbf{x}}^\lambda) + \lambda r_{\Delta x}(\breve{\mathbf{p}}^\lambda, \breve{\mathbf{x}}^\lambda). \tag{2.4}$$

The homotopy-minimum functions, $\breve{\rho}(\lambda)$, $\breve{r}_{\mathrm{y}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$, are useful because they admit conservative quantities, which allow me to gauge the progress and accuracy of my minimization technique. I discuss details in Section 2.8.

2.3 Defining a Measure of Data Fitting, $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$

As an example and for use later, I define $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$. In doing so, I consider the sum of weighted squared differences as a measure of the difference between the j^{th} observable model state and the corresponding observed data values:

$$\sum_{k=1}^{n_t} w_{j,k}\big(\mathrm{y}_{j,k} - g_j(\mathbf{p}, x_{1,k}, x_{2,k}, \ldots, x_{n_x,k})\big)^2 = \sum_{k=1}^{n_t} w_{j,k}\big(\mathrm{y}_{j,k} - g_j(\mathbf{p},\mathbf{x}_k)\big)^2, \tag{2.5}$$

for some data-dependent weights $w_{j,k}$. To simultaneously measure the difference between all observable model states and all observed data values, I combine the sum of weighted squared differences, for all observable states, $j = 1, 2, \ldots, n_y$, into the single functional $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$. In doing so, to combine weighted squared differences on mixed scales with mixed units, I normalized each sum by the sum of weighted squared observed data values. To remove dependence on the number of observable states, I divide the normalized sum by the number of observable states. Thus,

$$r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) = \frac{1}{n_y}\sum_{j=1}^{n_y}\left(\frac{1}{\sum_{k=1}^{n_t} w_{j,k}\mathrm{y}_{j,k}^2}\sum_{k=1}^{n_t} w_{j,k}\big(\mathrm{y}_{j,k} - g_j(\mathbf{p},\mathbf{x}_k)\big)^2\right). \tag{2.6}$$

Without normalization, minimizing $\rho(\mathbf{p},\mathbf{x};\lambda)$ biases fitting toward data on a larger scale at the detriment of fitting data on a smaller scale. Normalization also removes dependence of $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$ on the number of data points. In cases where data-dependent weights, $w_{j,k}$, correct for disparities in scale and units, normalization standardizes the scale of $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$. To give the reader a sense of scale, for the homogeneous model, $g_j = 0$ for all $j \in \{1, 2, \ldots, n_y\}$, of nontrivial data, $r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) = 1$.

2.4 Defining a Measure of Satisfying a Numerical Solution, $r_{\Delta x}(\mathbf{p},\mathbf{x})$

As an example and for use later, I define $r_{\Delta x}(\mathbf{p},\mathbf{x})$. In doing so, I consider systems of first order ordinary differential equations that are linear in derivatives of x_i,

$$\frac{dx_i}{dt} = \bar{F}_i\left(t, p_1, p_2, \ldots, p_{n_p}, x_1, x_2, \ldots, x_{n_x}\right), \tag{2.7}$$

for $i \in \{1, 2, \ldots, n_x\}$. In terms of F_i as defined in equation (2.1a),

$$F_i = \frac{dx_i}{dt} - \bar{F}_i\left(t, p_1, p_2, \ldots, p_{n_p}, x_1, x_2, \ldots, x_{n_x}\right), \tag{2.8}$$

for $i \in \{1, 2, \ldots, n_x\}$. I also consider finite difference numerical methods, of the form

$$\begin{gathered}\Delta x_{i,k} = \\ F_{i,k}\big(\left\{\bar{F}_i(t_k, p_1, p_2, \ldots, p_{n_p}, x_{1,k}, x_{2,k}, \ldots, x_{n_x,k}) : k \in \mathcal{I}_\Delta\right\}\big) = F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}),\end{gathered} \tag{2.9}$$

where $x_{i,k}$ are the numerical solution values, $\Delta x_{i,k}$ is some method-dependent finite difference discretization of $\frac{dx_i}{dt}$ at time t_k, and $F_{i,k}$ are some method-dependent functions of $\bar{F}_i$ at time t_k, for all $i \in \{1, 2, \ldots, n_x\}$ and for all $k \in \mathcal{I}_\Delta$. In terms of $f_{i,k}$ as defined in equation (2.2),

$$f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) = \Delta x_{i,k} - F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}), \tag{2.10}$$

for all $i \in \{1, 2, \ldots, n_x\}$ and for all $k \in \mathcal{I}_\Delta$. For example, with the backward Euler method,

$$\begin{aligned}\Delta x_{i,k} &= \begin{cases} 0 & \text{if } k \in \{1\} \\ \dfrac{x_{i,k} - x_{i,k_-}}{t_k - t_{k_-}} & \text{if } k \in \mathcal{I}_\Delta \setminus \{1\}, \end{cases} \\ F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) &= \begin{cases} 0 & \text{if } k \in \{1\} \\ \bar{F}_i\left(t_k, \mathbf{p}, \mathbf{x}_k\right) & \text{if } k \in \mathcal{I}_\Delta \setminus \{1\}, \end{cases}\end{aligned} \tag{2.11}$$

where k_- is the index below k in $\mathcal{I}_\Delta$, for $i \in \{1, 2, \ldots, n_x\}$.

To be consistent in measure, scale, and units with $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ as defined in equation (2.6), I measure the difference between all $\Delta x_{i,k}$ and $F_{i,k}$ by the mean normalized sum of squared differences,

$$r_{\Delta x}(\mathbf{p}, \mathbf{x}) = \frac{1}{n_x} \sum_{i=1}^{n_x} \left(\frac{1}{\sum_{k \in \mathcal{I}_\Delta} (\Delta x_{i,k})^2} \sum_{k \in \mathcal{I}_\Delta} \left(\Delta x_{i,k} - F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})\right)^2 \right). \tag{2.12}$$

Normalizing by the sum of squared finite differences allows me to combine squared differences for variables on mixed scales with mixed units. Without normalization, minimizing $\rho(\mathbf{p}, \mathbf{x}; \lambda)$ biases fitting toward discretizations in more rapidly changing states at the detriment of fitting discretizations in more slowly changing states. Normalization also removes dependence of $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ on the number of points in the discretization, and dividing by the number of states removes dependence of $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ on the number of states. To give the reader a sense of scale, for the homogeneous model, $F_{i,k} = 0$ for all $i \in \{1, 2, \ldots, n_x\}$ and for all $k \in \mathcal{I}_\Delta$, of nonconstant state values, $r_{\Delta x}(\mathbf{p}, \mathbf{x}) = 1$. Thus, $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ is consistent in scale and units with $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ of equation (2.6). For systems of first order ordinary differential equations that are nonlinear in derivatives of x_i, or for numerical methods other than finite difference methods, $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ should

be normalized similarly.

2.4.1 Inclusion of a Smoothing Penalty in $r_{\Delta x}(\mathbf{p}, \mathbf{x})$

Observable states are often the combination of multiple unobservable states. As such, it is possible for observable states to fit observed data with jagged underlying state values. Because of normalization by sums of squared finite differences in $r_{\Delta x}(\mathbf{p}, \mathbf{x})$, jagged state values may lead to relatively small values $r_{\Delta x}(\mathbf{p}, \mathbf{x})$. As $\lambda \to 1^-$, jagged state values are dampened out in the minimization of $\rho(\mathbf{p}, \mathbf{x}; \lambda) = (1 - \lambda) r_{\mathrm{y}}(\mathbf{p}, \mathbf{x}) + \lambda r_{\Delta x}(\mathbf{p}, \mathbf{x})$, as $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ approaches a value of 0, which occurs if and only if state values approach a numerical solution. However, for small λ, jagged state values may lead to smaller values of $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ than would less jagged state values that are closer to a numerical solution. Jagged state values do not conform to the dynamic structure of the differential equation model and, thus, do not meaningfully guide the minimization of $\rho(\mathbf{p}, \mathbf{x}; \lambda)$ for larger values of λ. To avoid jagged state values when minimizing $\rho(\mathbf{p}, \mathbf{x}; \lambda)$ for small values of λ, I incorporate multiplicative smoothing penalties, $s_i(\mathbf{x})$, into $r_{\Delta x}(\mathbf{p}, \mathbf{x})$:

$$r_{\Delta x}(\mathbf{p}, \mathbf{x}) = \frac{1}{n_x} \sum_{i=1}^{n_x} \left(\frac{s_i(\mathbf{x})}{\sum_{k \in \mathcal{I}_\Delta} (\Delta x_{i,k})^2} \sum_{k \in \mathcal{I}_\Delta} \left(\Delta x_{i,k} - F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) \right)^2 \right), \tag{2.13a}$$

$$s_i(\mathbf{x}) = \alpha_i + \beta_i \left(\frac{4 \sum_{k \in \mathcal{I}_\Delta \setminus \{1, n_t\}} \left(x_{i,k_-} - 2x_{i,k} + x_{i,k_+} \right)^2}{\sum_{k \in \mathcal{I}_\Delta \setminus \{1, n_t\}} \left(x_{i,k_+} - x_{i,k_-} \right)^2} \right)^{\gamma_i}, \tag{2.13b}$$

where k_- and k_+ are the indices below and above k in $\mathcal{I}_\Delta$; $(x_{i,k_-} - 2x_{i,k} + x_{i,k_+})^2$ is a measure of roughness around $x_{i,k}$, the squared difference between the forward difference centered at $x_{i,k}$, $x_{i,k_+} - x_{i,k}$, and the backward difference centered at $x_{i,k}$, $x_{i,k} - x_{i,k_-}$; $(x_{i,k_+} - x_{i,k_-})^2/4$ is a normalization measure of differences centered at $x_{i,k}$, the squared mean value of the forward difference centered at $x_{i,k}$ and the backward difference centered at $x_{i,k}$; and $\alpha_i > 0$, $\beta_i \geq 0$, and $\gamma_i \geq 0$ are parameters that are chosen to ensure that state values do not jaggedly deviate from the dynamic structure of the model and to set the scale of the smoothing penalty. To give a sense of scale, I note that $s_i(\mathbf{x}) = 0$ with $\alpha_i = 0$, $\beta_i = 1$, and $\gamma_i = 1$ for $x_{i,k}$ evenly spaced along a line with nonzero slope, and $s_i(\mathbf{x})$ ranges from around 10 to 15 with $\alpha_i = 0$, $\beta_i = 1$, and $\gamma_i = 1$ for $x_{i,k}$ randomly sampled from the standard uniform distribution or the standard normal distribution with $k \in \{1, 2, \ldots, 10^3\}$. Thus, as an example, choosing $\alpha_i = 1$, $\beta_i = 10^2$, and $\gamma_i = 2$ for all $i \in \{1, 2, \ldots, n_x\}$ would insignificantly modify $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ when state values are close to colinear and would strongly penalize $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ when state values are close to random. I note that, as $\alpha_i > 0$ for all $i \in \{1, 2, \ldots, n_x\}$, $r_{\Delta x}(\mathbf{p}, \mathbf{x}) = 0$ if and only if $\Delta x_{i,k} = F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ for all $i \in \{1, 2, \ldots, n_x\}$ and all $k \in \mathcal{I}_\Delta$. As such, the inclusion of multiplicative smoothing penalties in $r_{\Delta x}(\mathbf{p}, \mathbf{x})$, as defined in equation (2.13), does not alter the parameters and state values that minimize $\rho(\mathbf{p}, \mathbf{x}; \lambda)$ as $\lambda \to 1^-$.

2.5 A Concrete Example of $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$ and $r_{\Delta x}(\mathbf{p},\mathbf{x})$ Using a Model of FRAP

I provide a concrete example of $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$ and $r_{\Delta x}(\mathbf{p},\mathbf{x})$ for a simple model of fluorescence recovery after photobleaching (FRAP). For fluorescence intensity $x_1(t)$, recovery-level parameter p_1, timescale parameter p_2, and observable state y_1,

$$\frac{dx_1}{dt} = \frac{p_1 - x_1}{p_2}, \tag{2.14a}$$

$$y_1 = g_1(p_1, p_2, x_1) = x_1. \tag{2.14b}$$

I consider some observed data values $\mathrm{y}_{1,k}$ measured at times t_k, for $k \in \{1, 2, \ldots, n_t\}$, and discrete state values $x_{1,k}$ on an unrefined discretization grid, $k \in \mathcal{I}_\Delta = \{1, 2, \ldots, n_t\}$. Thus, for $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$ as defined in equation 2.6 with unitary weights and $r_{\Delta x}(\mathbf{p},\mathbf{x})$ as defined in equation 2.12 using the backward Euler discretization as defined in equation 2.11,

$$r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) = \frac{1}{\sum_{k=1}^{n_t} \mathrm{y}_{1,k}^2} \sum_{k=1}^{n_t} \left(\mathrm{y}_{1,k} - x_{1,k}\right)^2, \tag{2.15a}$$

$$r_{\Delta x}(\mathbf{p},\mathbf{x}) = \left(\sum_{k=2}^{n_t} \left(\frac{x_{1,k} - x_{1,k-1}}{t_k - t_{k-1}}\right)^2\right)^{-1} \sum_{k=2}^{n_t} \left(\frac{x_{1,k} - x_{1,k-1}}{t_k - t_{k-1}} - \frac{p_1 - x_{1,k}}{p_2}\right)^2. \tag{2.15b}$$

2.6 Extending the Homotopy on Refined Discretization Grids

For $\rho(\mathbf{p},\mathbf{x};\lambda)$ with small λ, on a refined discretization grid, where $\mathcal{I}_\Delta \neq \{1, 2, \ldots, n_t\}$, deviations in observable state values from observed data values carry a strong penalty at grid points with indices in $\{1, 2, \ldots, n_t\}$ and no penalty at grid points with indices in $\mathcal{I}_\Delta \setminus \{1, 2, \ldots, n_t\}$. Thus, the state values that minimize $\rho(\mathbf{p},\mathbf{x};\lambda)$ with small λ may admit observable state values that vary dramatically across adjacent indices in $\{1, 2, \ldots, n_t\}$ and $\mathcal{I}_\Delta \setminus \{1, 2, \ldots, n_t\}$. As λ increases from 0 to 1, the state values that minimize $\rho(\mathbf{p},\mathbf{x};\lambda)$ increasingly inherit smooth structure from the solution to differential equation system (2.1a). Smoothness transfers from state values to observable state values through the observation functions, g_j. Thus, the state values that minimize $\rho(\mathbf{p},\mathbf{x};\lambda)$ with λ near 1 admit observable state values that vary smoothly across adjacent indices in $\{1, 2, \ldots, n_t\}$ and $\mathcal{I}_\Delta \setminus \{1, 2, \ldots, n_t\}$. As such, on a refined discretization grid, the parameters and state values that minimize $\rho(\mathbf{p},\mathbf{x};\lambda)$ for small values of λ may not meaningfully guide the minimization of $\rho(\mathbf{p},\mathbf{x};\lambda)$ for larger values of λ. To address this, I extend $\rho(\mathbf{p},\mathbf{x};\lambda)$ to penalize deviations in observable state values from interpolated data values at grid points with indices in $\mathcal{I}_\Delta \setminus \{1, 2, \ldots, n_t\}$.

Using some interpolation method, I generate interpolated data, $\hat{\mathrm{y}}_{j,k}$ for $j \in \{1, 2, \ldots, n_y\}$, at grid points with indices in $\mathcal{I}_{\hat{\mathrm{y}}} = \mathcal{I}_\Delta \setminus \{1, 2, \ldots, n_t\}$ from observed data values with indices in $\{1, 2, \ldots, n_t\}$. Using some measure of the difference between observable model states and

interpolated data values, I can define the functional $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x})$ with the properties that (i) $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) \geq 0$, (ii) $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) = 0$ if and only if $\mathcal{I}_\Delta = \{1,2,\ldots,n_t\}$ or $g_j(\mathbf{p},x_{1,k},\ldots,x_{n_x,k}) = \hat{\mathrm{y}}_{j,k}$ for all $j \in \{1,2,\ldots,n_y\}$ and for all $k \in \mathcal{I}_{\hat{\mathrm{y}}}$, and (iii) $r_{\hat{\mathrm{y}}}(\mathbf{p}_1,\mathbf{x}_1) < r_{\hat{\mathrm{y}}}(\mathbf{p}_2,\mathbf{x}_2)$ implies that $(\mathbf{p}_1,\mathbf{x}_1)$ gives a better fit to the interpolated data than does $(\mathbf{p}_2,\mathbf{x}_2)$. I describe the construction of $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x})$ under a normalized least-squares measure in Section 2.6.1. I extend $\rho(\mathbf{p},\mathbf{x};\lambda)$ to include $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x})$ with proportionality value $(1-\lambda)^2$:

$$\begin{aligned} r(\mathbf{p},\mathbf{x};\lambda) = \rho(\mathbf{p},\mathbf{x};\lambda) + (1-\lambda)^2 r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) = \\ (1-\lambda) r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) + (1-\lambda)^2 r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) + \lambda r_{\Delta x}(\mathbf{p},\mathbf{x}), \end{aligned} \tag{2.16}$$

a homotopy between $r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) + r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x})$ and $r_{\Delta x}(\mathbf{p},\mathbf{x})$. The proportionality value of $(1-\lambda)^2$ on $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x})$ incrementally decreases the weighting of $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x})$ relative to $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$, as λ increases from 0 to 1 in $r(\mathbf{p},\mathbf{x};\lambda)$. I show in Section B.1 that if $(\check{\mathbf{p}},\check{\mathbf{x}})$ minimizes $\rho(\mathbf{p},\mathbf{x};\lambda)$ as $\lambda \to 1^-$ then $(\check{\mathbf{p}},\check{\mathbf{x}})$ also minimizes $r(\mathbf{p},\mathbf{x};\lambda)$ as $\lambda \to 1^-$. Thus, the inclusion of $(1-\lambda)^2 r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x})$ in $\rho(\mathbf{p},\mathbf{x};\lambda)$ penalizes deviations in observable state values from interpolated data values at smaller values of λ, but does not alter the parameters and state values that minimize $\rho(\mathbf{p},\mathbf{x};\lambda)$ as $\lambda \to 1^-$.

2.6.1 Defining a Measure of Interpolated Data Fitting, $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x})$

As an example and for use later, I define $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x})$ following the form of $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$ in equation (2.6). To do so, using some interpolation method, I generate interpolated data-dependent weights, $\hat{w}_{j,k}$ for $j \in \{1,2,\ldots,n_y\}$, at grid points with indices in $\mathcal{I}_{\hat{\mathrm{y}}}$ from data-dependent weights with indices in $\{1,2,\ldots,n_t\}$. To be consistent in measure and scale with $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$ as defined in equation (2.6), I measure the difference between observable state values and interpolated data values at grid points with indices in $\mathcal{I}_{\hat{\mathrm{y}}}$ by the mean normalized sum of squared differences:

$$r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) = \frac{1}{n_y}\sum_{j=1}^{n_y}\left(\frac{\hat{\sigma}}{\sum_{k\in\mathcal{I}_{\hat{\mathrm{y}}}}\hat{w}_{j,k}\hat{\mathrm{y}}_{j,k}^2}\sum_{k\in\mathcal{I}_{\hat{\mathrm{y}}}}\hat{w}_{j,k}\big(\hat{\mathrm{y}}_{j,k} - g_j(\mathbf{p},x_{1,k},\ldots,x_{n_x,k})\big)^2\right), \tag{2.17}$$

where $\hat{\sigma} > 0$ is a scaling parameter that is chosen to set the weighting of $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x})$ relative to $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$ as $\lambda \to 0^+$. If $\mathcal{I}_\Delta = \{1,2,\ldots,n_t\}$, then I define $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) = 0$, and $r(\mathbf{p},\mathbf{x};\lambda)$ reduces to $\rho(\mathbf{p},\mathbf{x};\lambda)$.

2.7 Optimization Using Overlapping-Niche Descent

To synergistically minimize $r(\mathbf{p},\mathbf{x};\lambda)$ over an array of λ values, I implement overlapping-niche descent, a genetic algorithm directed by gradient-based descent, which is isomorphic to the dynamics of an evolving ecological population that competes for multiple food sources in a single environment, where an individual that successfully competes for one food source may successfully compete for a similar food source. In overlapping-niche descent, a unique value of

$\lambda \in (0, 1)$ defines a niche, and a set of λ values spanning $(0, 1)$ defines the set of niches in the environment. Each niche supports a certain number of individuals, where each individual is represented by a set of parameters and state values. As in other genetic algorithms, individuals reproduce, with crossover and mutation, to generate new individuals and variability within the parameter-state value search space. The likelihood of optimizing a function by random probing decreases with an increasing number of variables, and $r(\mathbf{p}, \mathbf{x}; \lambda)$ is a functional of many variables. Thus, to accelerate optimization, after reproduction, individuals undergo gradient-based descent. After descent, through selection, each niche sustains the individuals with the lowest values of $r(\mathbf{p}, \mathbf{x}; \lambda)$. Selection acts across niches, allowing individuals to spread from one niche to others, for a cooperative transfer of information from data to the optimal data-fitting numerical solution during optimization. I discuss details of overlapping-niche descent in Section C. The process of overlapping-niche descent is depicted in Figure 2.1.

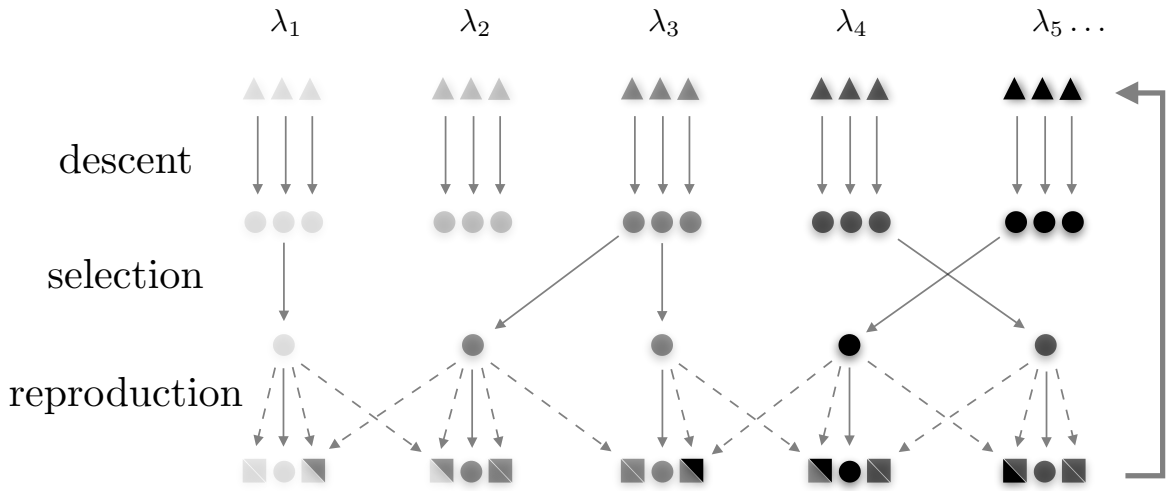


Figure 2.1: Overlapping-Niche Descent. The parameters and state values that minimize $r(\mathbf{p}, \mathbf{x}; \lambda)$ as $\lambda \to 1^-$ are those of the optimal data-fitting numerical solution. Overlapping-niche descent synergistically minimizes $r(\mathbf{p}, \mathbf{x}; \lambda)$ over a broad range of λ values, $\lambda_1, \lambda_2, \ldots$, to more robustly minimize $r(\mathbf{p}, \mathbf{x}; \lambda)$ as $\lambda \to 1^-$. In overlapping-niche descent, each value of λ defines a niche. In each niche, $r(\mathbf{p}, \mathbf{x}; \lambda)$ is locally minimized for a set of initial parameters and state values (descent). From the full set of parameters and state values, the parameters and state values with the lowest values of $r(\mathbf{p}, \mathbf{x}; \lambda)$ are retained in each niche (selection). Then, a new set of parameters and state values are generated in each niche from the full set of retained parameters and state values (reproduction). After selection and reproduction, descent occurs again. Initial points are shown with triangles, local minimums are shown with circles, and newly generated points are shown with squares. Gradation from light gray to black is shown to emphasize the transfer of information across niches during optimization.

Although similar in name, my overlapping-niche genetic algorithm, which cooperatively optimizes over a range of similar problems, differs from multi-niche genetic algorithms, which optimize a single problem to find multiple modes. Overlapping niche-descent is less similar in name, but more similar in character to smooth functional tempering [8], a Bayesian, dynamic

colocation method. Smooth functional tempering employs parallel MCMC (Markov Chain Monte Carlo) chains, over a range of derivative-matching penalty weights. During sampling, parallel chains may exchange parameters to more robustly sample posterior probability distributions in chains with large derivative-matching penalty weights. Ultimately, smooth functional tempering approximates the posterior probability distribution of a dynamic collocation basis with some large derivative-matching penalty weight.

2.8 Properties of the Homotopy and Inspection of Overlapping-Niche Descent

For $\lambda \in (0, 1)$, the parameters and state values that minimize $r(\mathbf{p}, \mathbf{x}; \lambda)$, $\breve{\mathbf{p}}^{\lambda}$ and $\breve{\mathbf{x}}^{\lambda}$, allow me to define useful functions, $\breve{r}(\lambda)$, $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$, as follows:

$$\begin{aligned}\breve{r}(\lambda) = (1-\lambda)\breve{r}_{\mathrm{y}}(\lambda) + (1-\lambda)^2\breve{r}_{\hat{\mathrm{y}}}(\lambda) + \lambda\breve{r}_{\Delta x}(\lambda) = \\ (1-\lambda)r_{\mathrm{y}}(\breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda}) + (1-\lambda)^2 r_{\hat{\mathrm{y}}}(\breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda}) + \lambda r_{\Delta x}(\breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda}).\end{aligned} \tag{2.18}$$

The homotopy-minimum functions, $\breve{r}(\lambda)$, $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$, are useful because they admit conservative quantities, which allow me to gauge the progress and accuracy of overlapping-niche descent.

From Theorem 1 in Appendix B,

$$\lim_{\lambda \to 0^+} \breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda} = \arg\min \big(r_{\Delta x}(\mathbf{p}, \mathbf{x}) : r_{\mathrm{y}}(\mathbf{p}, \mathbf{x}) = 0, r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x}) = 0\big), \tag{2.19}$$

$$\lim_{\lambda \to 1^-} \breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda} = \arg\min \big(r_{\mathrm{y}}(\mathbf{p}, \mathbf{x}) : r_{\Delta x}(\mathbf{p}, \mathbf{x}) = 0\big). \tag{2.20}$$

Equation (2.19) states that the parameters and state values that minimize $r(\mathbf{p}, \mathbf{x}; \lambda)$ as $\lambda \to 0^+$ are those closest to a numerical solution given that observable state values perfectly fit data; equation (2.20) states that the parameters and state values that minimize $r(\mathbf{p}, \mathbf{x}; \lambda)$ as $\lambda \to 1^-$ are those of the numerical solution that fits observed data best. Thus, $\lim_{\lambda \to 0^+} \breve{r}_{\Delta x}(\lambda)$ is a measure of how well data can satisfy a numerical solution, and $\lim_{\lambda \to 1^-} \breve{r}_{\mathrm{y}}(\lambda)$ is a measure of how well a numerical solution can fit observed data. Generally, I am interested in finding $\lim_{\lambda \to 1^-} \breve{\mathbf{p}}^{\lambda}$ and $\lim_{\lambda \to 1^-} \breve{\mathbf{x}}^{\lambda}$. However, $\lim_{\lambda \to 0^+} \breve{\mathbf{p}}^{\lambda}$ and $\lim_{\lambda \to 0^+} \breve{\mathbf{x}}^{\lambda}$ are also informative, as they show how badly data fails to be a numerical solution, and may thus provide insight into measurement error, model inadequacies, and potential model improvements.

From Theorem 3 in Appendix B, if $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ are differentiable at $\lambda \in (0, 1)$, then

$$(1-\lambda)\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda} + (1-\lambda)^2\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda} + \lambda\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda} = 0. \tag{2.21}$$

Thus, changes in $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ with respect to λ are coupled. If $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and

$\breve{r}_{\Delta x}(\lambda)$ are differentiable at all but a finite number of points in $(0,1)$, then from Theorem 4 in Appendix B,

$$2\int_0^1 \breve{r}(\lambda)d\lambda = \int_0^1 \breve{r}_{\mathrm{y}}(\lambda)d\lambda + \int_0^1 (1-\lambda^2)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda = \int_0^1 \breve{r}_{\Delta x}(\lambda)d\lambda - \int_0^1 (1-\lambda)^2\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda. \tag{2.22}$$

Equation (2.22) defines a conservation in the coupled changes of $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ across $\lambda \in (0,1)$. In overlapping-niche descent, I minimize $r(\mathbf{p},\mathbf{x};\lambda)$ over an array of λ values in $(0,1)$ to find $\tilde{\mathbf{p}}^{\lambda}$ and $\tilde{\mathbf{x}}^{\lambda}$, approximations of $\breve{\mathbf{p}}^{\lambda}$ and $\breve{\mathbf{x}}^{\lambda}$, which allow me to define the functions $\tilde{r}(\lambda)$, $\tilde{r}_{\mathrm{y}}(\lambda)$, $\tilde{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\tilde{r}_{\Delta x}(\lambda)$ such that

$$\tilde{r}(\lambda) = (1-\lambda)\tilde{r}_{\mathrm{y}}(\lambda) + (1-\lambda)^2\tilde{r}_{\hat{\mathrm{y}}}(\lambda) + \lambda\tilde{r}_{\Delta x}(\lambda) = (1-\lambda)r_{\mathrm{y}}(\tilde{\mathbf{p}}^{\lambda},\tilde{\mathbf{x}}^{\lambda}) + (1-\lambda)^2 r_{\hat{\mathrm{y}}}(\tilde{\mathbf{p}}^{\lambda},\tilde{\mathbf{x}}^{\lambda}) + \lambda r_{\Delta x}(\tilde{\mathbf{p}}^{\lambda},\tilde{\mathbf{x}}^{\lambda}). \tag{2.23}$$

I can determine how well $\tilde{r}_{\mathrm{y}}(\lambda)$, $\tilde{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\tilde{r}_{\Delta x}(\lambda)$ satisfy conservation in coupled functional changes:

$$2\int_0^1 \tilde{r}(\lambda)d\lambda = \int_0^1 \tilde{r}_{\mathrm{y}}(\lambda)d\lambda + \int_0^1 (1-\lambda^2)\tilde{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda = \int_0^1 \tilde{r}_{\Delta x}(\lambda)d\lambda - \int_0^1 (1-\lambda)^2\tilde{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda. \tag{2.24}$$

A failure to reasonably satisfy equation (2.24) indicates that $\tilde{r}_{\mathrm{y}}(\lambda)$, $\tilde{r}_{\hat{\mathrm{y}}}(\lambda)$, and/or $\tilde{r}_{\Delta x}(\lambda)$ differ significantly from $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and/or $\breve{r}_{\Delta x}(\lambda)$, implying that overlapping-niche descent has not been successfully or is incomplete.

Conservation in coupled functional changes, equation (2.22), relates $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ over a broad range of $\lambda \in (0,1)$. However, values of $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ for λ near 0 and λ near 1 do not significantly affect the values of integrals in equation (2.22). Thus, reasonably satisfying equation (2.24) reveals little about the coupled changes of $\tilde{r}_{\mathrm{y}}(\lambda)$, $\tilde{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\tilde{r}_{\Delta x}(\lambda)$ for λ near 0 and λ near 1. From Theorem 5 in Appendix B, if $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ are differentiable at all but a finite number of points in $(0,1)$, then

$$\lim_{\lambda\to 0^+} \breve{r}_{\Delta x}(\lambda) = \int_0^1 \frac{1}{\lambda^2}\breve{r}_{\mathrm{y}}(\lambda)d\lambda + \int_0^1 \frac{1-\lambda^2}{\lambda^2}\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda, \tag{2.25a}$$

$$\lim_{\lambda\to 1^-} \breve{r}_{\mathrm{y}}(\lambda) = \int_0^1 \frac{1}{(1-\lambda)^2}\breve{r}_{\Delta x}(\lambda)d\lambda - \int_0^1 \breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda. \tag{2.25b}$$

Equation 2.25 defines integral representations of limit values, with $\lim_{\lambda\to 0^+} \breve{r}_{\Delta x}(\lambda)$ defined entirely in terms of $\breve{r}_{\mathrm{y}}(\lambda)$ and $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\lim_{\lambda\to 1^-} \breve{r}_{\mathrm{y}}(\lambda)$ defined entirely in terms of $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$ and $\breve{r}_{\Delta x}(\lambda)$. I can determine how well $\tilde{r}_{\mathrm{y}}(\lambda)$, $\tilde{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\tilde{r}_{\Delta x}(\lambda)$ satisfy the integral representations

of limit values:

$$\lim_{\lambda\to 0^+} \tilde{r}_{\Delta x}(\lambda) = \int_0^1 \frac{1}{\lambda^2}\tilde{r}_{\mathrm{y}}(\lambda)d\lambda + \int_0^1 \frac{1-\lambda^2}{\lambda^2}\tilde{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda, \tag{2.26a}$$

$$\lim_{\lambda\to 1^-} \tilde{r}_{\mathrm{y}}(\lambda) = \int_0^1 \frac{1}{(1-\lambda)^2}\tilde{r}_{\Delta x}(\lambda)d\lambda - \int_0^1 \tilde{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda. \tag{2.26b}$$

A failure to reasonably satisfy equation (2.26) indicates that $\tilde{r}_{\mathrm{y}}(\lambda)$, $\tilde{r}_{\hat{\mathrm{y}}}(\lambda)$, and/or $\tilde{r}_{\Delta x}(\lambda)$ differ significantly from $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and/or $\breve{r}_{\Delta x}(\lambda)$, implying that overlapping-niche descent has not been successfully or is incomplete.

I note that $r(\mathbf{p}, \mathbf{x}; \lambda)$ reduces to $\rho(\mathbf{p}, \mathbf{x}; \lambda)$ when $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x}) = 0$, and thus, the aforementioned properties of $r(\mathbf{p}, \mathbf{x}; \lambda)$ apply to $\rho(\mathbf{p}, \mathbf{x}; \lambda)$ with $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x}) = 0$.

Chapter 3

Testing the Homotopy-Minimization Method for Parameter Estimation in Differential Equations

3.1 Introduction

In Chapter 2, I developed a method that allowed me to calculate the optimal data-fitting numerical solution and its parameters for a differential equation model without using numerical integration. Additionally, I showed that my method admits conservation principles and integral representations that allow me to gauge the accuracy of my optimization. In this chapter, I test my method using a system of first order ordinary differential equations, a system of second order ordinary differential equations, and a system of partial differential equations. In doing so, I compare the performance of my method to that of an analogous numerical-integration-based method, explore how my method can inform modeling insufficiencies and potential model improvements, and expound how conservation principles and integral representations in my method gauge the accuracy of my optimization in practice.

As discussed in Section 1.2, the Min System, consisting of three proteins, MinD, MinE, and MinC, regulates the site of cell division in *Escherichia coli* [15]. Experimentally, MinD and MinE show interesting behaviors, such as emerging pole-to-pole oscillations in cells *in vivo* [60] and traveling waves and spiral waves on supported lipid bilayers *in vitro* [45]. The Bonny model [4], which models the interactions of MinD and MinE, admits solutions with behaviors that are qualitatively similar to the dynamic patternings of MinD and MinE that are observed in experiments [4]. I choose the Bonny model to test my method because of its biological relevance, and because in different contexts it manifests as a first order ordinary differential equation system, a second order ordinary differential equation system, and a system of partial differential equations.

3.2 A Model for MinD and MinE Interactions by Bonny *et al* (2013)

The Bonny model consists of five states, c_D, c_E, c_d, c_{de}, and c_e, corresponding to concentrations of bulk MinD, bulk MinE, membrane bound MinD, membrane bound MinD-MinE complex, and

membrane bound MinE respectively. I describe the biology behind the Bonny model and its formulation in detail in Section 4.3.2. I focus on a simplified version of the Bonny model with a large, well mixed bulk, such that MinD and MinE bulk concentrations, c_D and c_E, are constant:

$$\frac{\partial c_d}{\partial t} = c_D(\omega_D + \omega_{dD}c_d)(c_{\max} - c_d - c_{de})/c_{\max} - \omega_E c_E c_d - \omega_{ed}c_e c_d + D_d \nabla c_d, \tag{3.1a}$$

$$\frac{\partial c_{de}}{\partial t} = \omega_E c_E c_d + \omega_{ed}c_e c_d - (\omega_{de,m} + \omega_{de,c})c_{de} + D_{de}\nabla c_{de}, \tag{3.1b}$$

$$\frac{\partial c_e}{\partial t} = \omega_{de,m}c_{de} - \omega_{ed}c_e c_d - \omega_e c_e + D_e \nabla c_e, \tag{3.1c}$$

with parameters ω_D, ω_{dD}, ω_E, ω_{ed}, $\omega_{de,m}$, $\omega_{de,c}$, ω_e, $c_{\max}$, D_d, D_{de}, and D_e and observable states MinD $= c_d + c_{de}$ and MinE $= c_{de} + c_e$. c_d, c_{de}, c_e, MinD, and MinE are measured in μm^{-2}. Values and definitions of parameters and constants for the Bonny model in an *in vitro* context are shown in Table 3.1.

	value	definition
D_d	$3.00 \cdot 10^{-1}\ \mu m^2\ s^{-1}$	diffusion coefficient of c_d
D_{de}	$3.00 \cdot 10^{-1}\ \mu m^2\ s^{-1}$	diffusion coefficient of c_{de}
D_e	$1.80 \cdot 10^{0}\ \mu m^2\ s^{-1}$	diffusion coefficient of c_e
c_D	$4.80 \cdot 10^{2}\ \mu m^{-3}$	bulk concentration of MinD
c_E	$7.00 \cdot 10^{2}\ \mu m^{-3}$	bulk concentration of MinE
$c_{\max}$	$2.75 \cdot 10^{4}\ \mu m^{-2}$	maximum value of $c_d + c_{de}$
ω_D	$5.00 \cdot 10^{-4}\ \mu m\ s^{-1}$	rate of the reaction $c_D \to c_d$
ω_E	$1.36 \cdot 10^{-4}\ \mu m^3\ s^{-1}$	rate of the reaction $c_E + c_d \to c_{de}$
ω_{dD}	$3.18 \cdot 10^{-3}\ \mu m^3\ s^{-1}$	rate of the reaction $c_D + c_d \to 2c_d$
$\omega_{de,c}$	$1.60 \cdot 10^{-1}\ s^{-1}$	rate of the reaction $c_{de} \to c_D + c_E$
$\omega_{de,m}$	$2.52 \cdot 10^{0}\ s^{-1}$	rate of the reaction $c_{de} \to c_D + c_e$
ω_e	$5.00 \cdot 10^{-1}\ s^{-1}$	rate of the reaction $c_e \to c_E$
ω_{ed}	$4.90 \cdot 10^{-3}\ \mu m^2\ s^{-1}$	rate of the reaction $c_d + c_e \to c_{de}$

Table 3.1: Values and definitions of parameters and constants in the Bonny model. Values are taken from the set of *in vitro* parameters in [4].

In the case of spatial homogeneity, where $\nabla c_d = 0$, $\nabla c_{de} = 0$, and $\nabla c_e = 0$, the Bonny model reduces to a system of first order ordinary differential equations:

$$\frac{dc_d}{dt} = c_D(\omega_D + \omega_{dD}c_d)(c_{\max} - c_d - c_{de})/c_{\max} - \omega_E c_E c_d - \omega_{ed}c_e c_d, \tag{3.2a}$$

$$\frac{dc_{de}}{dt} = \omega_E c_E c_d + \omega_{ed}c_e c_d - (\omega_{de,m} + \omega_{de,c})c_{de}, \tag{3.2b}$$

$$\frac{dc_e}{dt} = \omega_{de,m}c_{de} - \omega_{ed}c_e c_d - \omega_e c_e. \tag{3.2c}$$

The Bonny model admits traveling wave solutions. In the traveling wave coordinate system, $z = x - st$, with spatial location x, time t, and nonzero traveling wave velocity s, where

$c_d(x,t) = c_d(z)$, $c_{de}(x,t) = c_{de}(z)$, and $c_e(x,t) = c_e(z)$, the Bonny model (3.1) reduces to a system of second order ordinary differential equations:

$$\frac{dc_d}{dz} = -\frac{1}{s}\left(c_D(\omega_D + \omega_{dD}c_d)(c_{\max} - c_d - c_{de})/c_{\max} - \omega_E c_E c_d - \omega_{ed}c_e c_d + D_d\frac{d^2c_d}{dz^2}\right), \quad (3.3a)$$

$$\frac{dc_{de}}{dz} = -\frac{1}{s}\left(\omega_E c_E c_d + \omega_{ed}c_e c_d - (\omega_{de,m} + \omega_{de,c})c_{de} + D_{de}\frac{d^2c_{de}}{dz^2}\right), \quad (3.3b)$$

$$\frac{dc_e}{dz} = -\frac{1}{s}\left(\omega_{de,m}c_{de} - \omega_{ed}c_e c_d - \omega_e c_e + D_e\frac{d^2c_e}{dz^2}\right). \quad (3.3c)$$

3.3 Synthetic Data Generation

Instead of fitting a form of the Bonny model to experimental data, I generate synthetic data from a numerical solution of the form of the Bonny model using the parameters in Table 3.1, and fit parameters in the form of the Bonny model to the synthetic data. This allows me to test my method within a controlled context, for a more concrete interpretation of my results.

The spatially homogeneous Bonny model (3.2) admits numerical solutions with oscillating pulses in MinD and MinE concentrations. To generate synthetic spatially-homogeneous data, I numerically solve the spatially homogeneous Bonny model (3.2) with the parameter values from Table 3.1 and small but non-zero initial conditions, $c_d(0) = 5.83\ \mu m^{-2}$, $c_{de}(0) = 1.34 \cdot 10^{-1}\ \mu m^{-2}$, and $c_e(0) = 1.58 \cdot 10^{-1}\ \mu m^{-2}$, to introduce some uncertainty in the values of initial conditions when fitting data. In doing so, I use the *MATLAB* ODE solver **ODE15s** with a relative error tolerance of 10^{-12} and an absolute error tolerance of 10^{-12}. I extract the synthetic spatially-homogeneous data by sampling the numerical solution every 0.5 s and calculating observable-state values. Synthetic spatially-homogeneous data is shown in Figure 3.1.

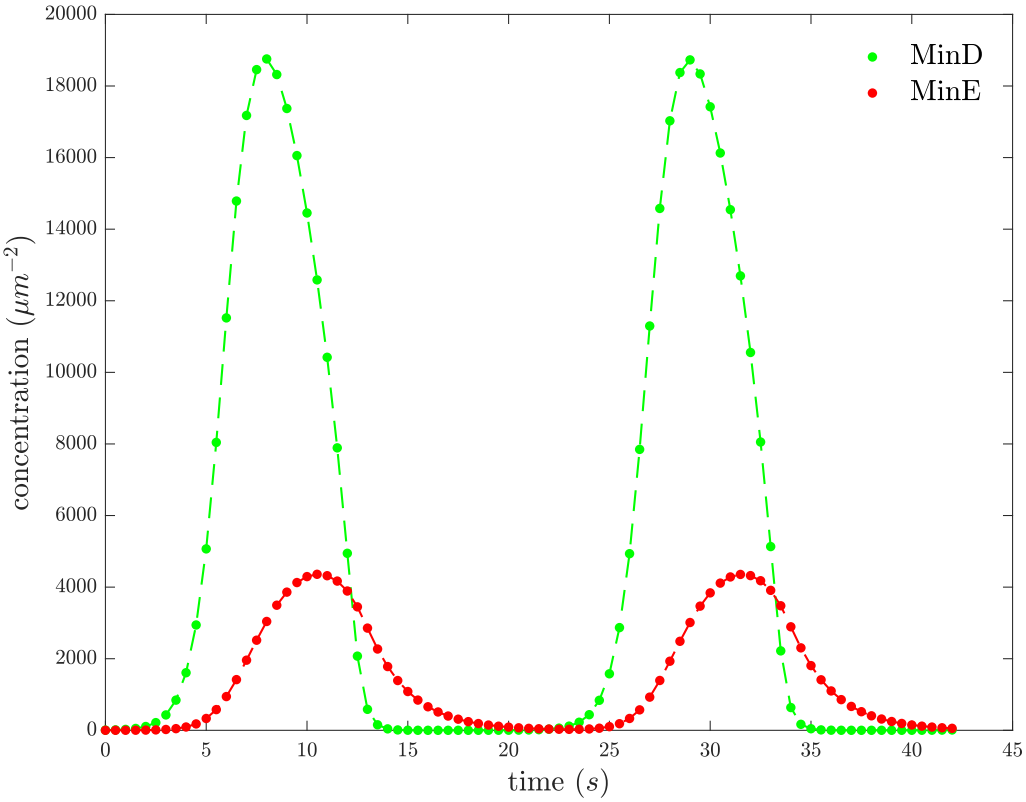


Figure 3.1: Synthetic spatially-homogeneous data. Data is generated by numerically solving the spatially homogeneous Bonny model (3.2) with the parameters from Table 3.1. Data is shown with points, and dashed lines are shown to emphasize the underlying pulse behavior.

To generate synthetic traveling-wave data, I construct a temporal pulse profile by numerically solving the spatially homogeneous Bonny model (3.2) with the parameter values from Table 3.1 and zero initial conditions. Then, I transform the temporal pulse profile into an initial pulse profile in space, and numerically evolve the pulse profile according to the Bonny model (3.1) with the parameter values from Table 3.1 and periodic boundary conditions. In doing so, I use the method of lines with a symmetric second order finite difference discretization of the Laplacian and RK4 time-stepping, on a grid with $1/8$ μm between spatial grid points and 10^{-3} s between temporal grid points. Over time, the pulse profile forms into a stable traveling wave profile, with measured traveling wave velocity of $s = -1.15$ μm s^{-1}. I extract the synthetic traveling-wave data by sampling the stable traveling wave profile every 0.5 μm and calculating observable-state values. Synthetic traveling-wave data is shown in Figure 3.2.

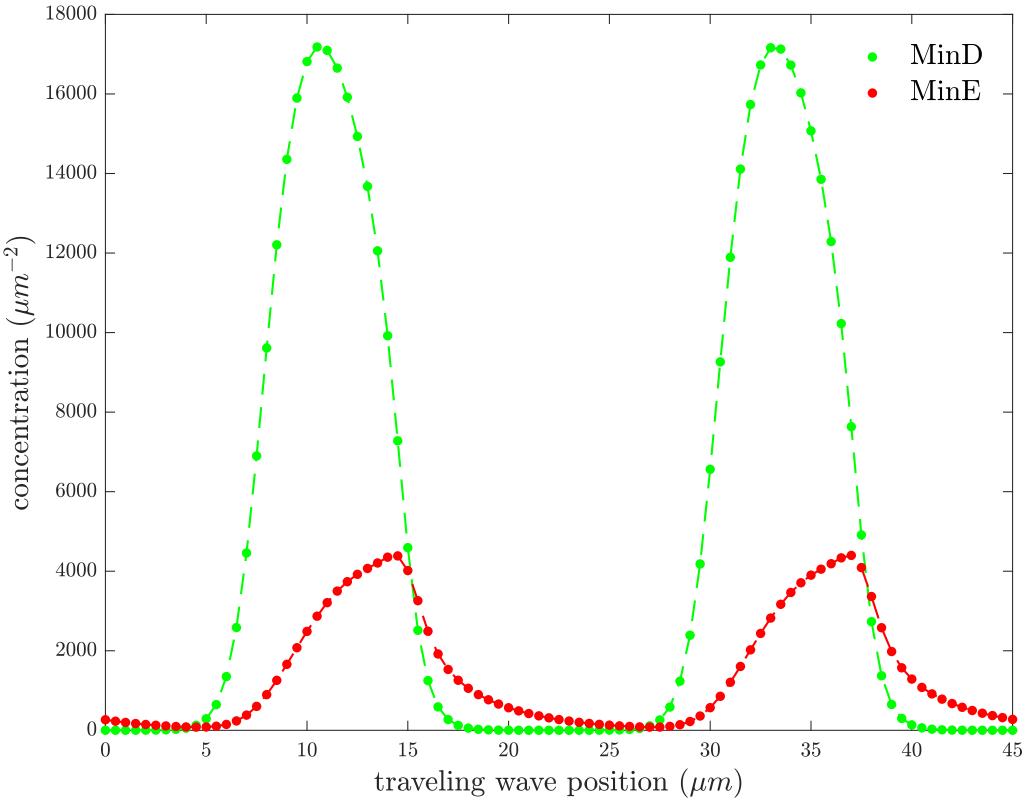


Figure 3.2: Synthetic traveling-wave data. Data is generated by numerically evolving a pulse with the full Bonny model (3.1) and the parameters from Table 3.1 until a stable traveling wave forms. Data is shown with points, and dashed lines are shown to emphasize the underlying traveling wave behavior.

The full Bonny model (3.1) demonstrates traveling wave emergence, the temporal evolution from a pulse profile into a stable traveling wave profile. I extract the synthetic traveling-wave-emergence data by sampling the numerical evolution of a pulse profile, as described above, every 0.5 μm and every 0.5 s for the first 15 s of its numerical evolution and calculating observable-state values. Synthetic traveling-wave-emergence data is shown in Figure 3.3.

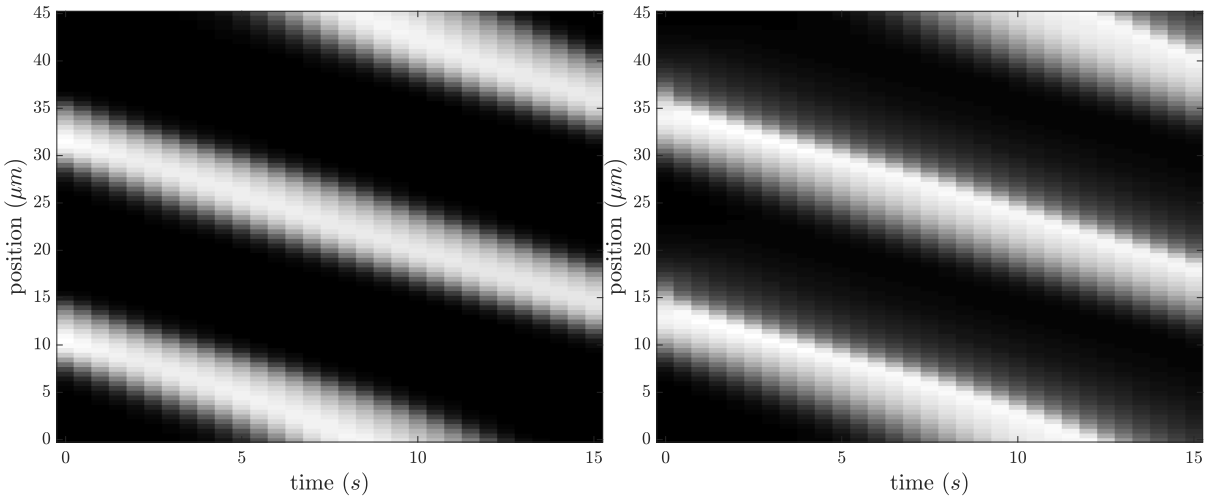


Figure 3.3: Synthetic traveling-wave-emergence data. Data is generated by numerically evolving a pulse with the full Bonny model (3.1) and the parameters from Table 3.1 for 15 s. MinD data is shown in (a), and MinE data is shown in (b). Gradation is from black, with a value of 0, to white, with a value of $1.9 \cdot 10^4$ in (a) and $4.5 \cdot 10^3$ in (b).

3.4 Details of Optimization Using Overlapping-Niche Descent

Here, I describe structural components of overlapping-niche descent for forms of the Bonny model. I describe details pertaining to the implementation of overlapping-niche descent in Section E.1.

3.4.1 Defining $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$, $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x})$, and $r_{\Delta x}(\mathbf{p}, \mathbf{x})$

Preliminarily, for consistency with previous notation, I define: $x_1 = c_d$, $x_2 = c_{de}$, and $x_3 = c_e$; $p_1 = D_d$, $p_2 = D_{de}$, $p_3 = D_e$, $p_4 = c_{\max}$, $p_5 = \omega_D$, $p_6 = \omega_E$, $p_7 = \omega_{dD}$, $p_8 = \omega_{de,c}$, $p_9 = \omega_{de,m}$, $p_{10} = \omega_e$, and $p_{11} = \omega_{ed}$; $y_1 = \mathrm{MinD}$ and $y_2 = \mathrm{MinE}$; and $g_1 = x_1 + x_2$ and $g_2 = x_2 + x_3$. Thus, $n_x = 3$, $n_p = 11$, and $n_y = 2$.

For the spatially homogeneous Bonny model (3.2), I define $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ as in equation (2.6), $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x})$ as in equation (2.17), and $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ as in equation (2.13a). In $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$, I use unitary data weights. In $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x})$, I set $\hat{\sigma} = 1$, and generate interpolated data and interpolated data weights using a piecewise cubic spline with not-a-knot end conditions. I discretize the spatially homogeneous Bonny model (3.2) using the backward Euler method, a method with first order

accuracy. Thus, in $r_{\Delta x}(\mathbf{p}, \mathbf{x})$,

$$\Delta x_{i,k} = \begin{cases} 0 & \text{if } k \in \{1\} \\ \dfrac{x_{i,k} - x_{i,k_-}}{\Delta t} & \text{if } k \in \mathcal{I}_\Delta \setminus \{1\}, \end{cases}$$
$$F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) = \begin{cases} 0 & \text{if } k \in \{1\} \\ \bar{F}_i\left(\mathbf{p}, x_{1,k}, x_{2,k}, \ldots, x_{n_x,k}\right) & \text{if } k \in \mathcal{I}_\Delta \setminus \{1\}, \end{cases} \tag{3.4}$$

where k_- is the index below k in $\mathcal{I}_\Delta$, Δt is the grid spacing in $\{t_k : k \in \mathcal{I}_\Delta\}$, and $\bar{F}_i$ is as defined in equation (2.7). In smoothing penalties, $s_i(\mathbf{x})$, of $r_{\Delta x}(\mathbf{p}, \mathbf{x})$, I set $\alpha_i = 1$, $\beta_i = 10^2$, and $\gamma_i = 2$, for all $i \in \{1, 2, \ldots, n_x\}$, to insignificantly modify $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ with a smooth set of state values and to strongly penalize $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ with a jagged set of state values.

For the traveling wave Bonny model (3.3), I define $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ as in equation (A.7a), $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x})$ as in equation (A.7b), and $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ as in equation (A.7c), with $t = z$. In $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$, I use unitary data weights. In $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x})$, I set $\hat{\sigma} = 1$, and generate interpolated data and interpolated data weights using a piecewise cubic spline with not-a-knot end conditions. I discretize the traveling wave Bonny model (3.3) using a central first order finite difference, a finite difference with second order accuracy, and a symmetric second order finite difference, a finite difference with second order accuracy. Thus, in $r_{\Delta x}(\mathbf{p}, \mathbf{x})$,

$$\Delta^1 x_{i,k} = \begin{cases} 0 & \text{if } k \in \{1, n_z\} \\ \dfrac{x_{i,k_+} - x_{i,k_-}}{2\Delta z} & \text{if } k \in \mathcal{I}_\Delta \setminus \{1, n_z\}, \end{cases}$$
$$\Delta^2 x_{i,k} = \begin{cases} 0 & \text{if } k \in \{1, n_z\} \\ \dfrac{x_{i,k_+} - 2x_{i,k} + x_{i,k_-}}{\Delta z^2} & \text{if } k \in \mathcal{I}_\Delta \setminus \{1, n_z\}, \end{cases}$$
$$F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) = \begin{cases} 0 & \text{if } k \in \{1, n_z\} \\ \bar{F}_i\left(\mathbf{p}, x_{1,k}, x_{2,k}, \ldots, x_{n_x,k}, \Delta^2 x_{i,k}\right) & \text{if } k \in \mathcal{I}_\Delta \setminus \{1, n_z\}, \end{cases} \tag{3.5}$$

where k_- and k_+ are the indices below and above k in $\mathcal{I}_\Delta$, Δz is the grid spacing in $\{z_k : k \in \mathcal{I}_\Delta\}$, and $\bar{F}_i$ is as defined in equation (A.3). As with the spatially homogeneous Bonny model, I set $\alpha_i = 1$, $\beta_i = 10^2$, and $\gamma_i = 2$ in smoothing penalties, $s_i(\mathbf{x})$, of $r_{\Delta x}(\mathbf{p}, \mathbf{x})$, for all $i \in \{1, 2, \ldots, n_x\}$, to insignificantly modify $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ with a smooth set of state values and to strongly penalize $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ with a jagged set of state values.

For the full Bonny model (3.1), I define $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ as in equation (A.14a), $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x})$ as in equation (A.14b), and $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ as in equation (A.14c), with u in time t and v in space s. In $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$, I use unitary data weights. In $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x})$, I set $\hat{\sigma} = 1$, and generate interpolated data and interpolated data weights using a two-dimensional piecewise cubic spline with not-a-knot end conditions. I discretize the full Bonny model (3.1) using a Simpson method first order finite difference in time, a finite difference with fourth order accuracy, and a symmetric second order

finite difference in space, a finite difference with second order accuracy. Thus, in $r_{\Delta x}(\mathbf{p}, \mathbf{x})$,

$$\Delta^{1,0} x_{i,k,l} = \begin{cases} 0 & \text{if } k \in \{1, n_t\} \text{ or } l \in \{1, n_s\} \\ \dfrac{x_{i,k_+,l} - x_{i,k_-,l}}{2\Delta t} & \text{if } (k,l) \in \mathcal{I}_{\Delta_t} \setminus \{1, n_t\} \times \mathcal{I}_{\Delta_s} \setminus \{1, n_s\}, \end{cases}$$

$$\Delta^{0,2} x_{i,k,l} = \begin{cases} 0 & \text{if } l \in \{1, n_s\} \\ \dfrac{x_{i,k,l_+} - 2x_{i,k,l} + x_{i,k,l_-}}{\Delta s^2} & \text{if } l \in \mathcal{I}_{\Delta_s} \setminus \{1, n_s\}, \end{cases}$$

$$F_{i,k,l}(\mathbf{t}, \mathbf{p}, \mathbf{x}) = \begin{cases} 0 & \text{if } k \in \{1, n_t\} \text{ or } l \in \{1, n_s\} \\ \displaystyle\sum_{m=-1}^{1} b_m \bar{F}_i \left(\mathbf{p}, \mathbf{x}_{k+m,l}, \Delta^{0,2} x_{i,k+m,l}\right) & \text{if } (k,l) \in \mathcal{I}_{\Delta_t} \setminus \{1, n_t\} \times \mathcal{I}_{\Delta_s} \setminus \{1, n_s\}, \end{cases} \tag{3.6}$$

where k_- and k_+ are the indices below and above k in $\mathcal{I}_{\Delta_t}$, l_- and l_+ are the indices below and above l in $\mathcal{I}_{\Delta_s}$, Δt is the grid spacing in $\{t_k : k \in \mathcal{I}_{\Delta_t}\}$, Δs is the grid spacing in $\{s_l : l \in \mathcal{I}_{\Delta_s}\}$, $b_{-1} = 1/6$, $b_0 = 4/6$, $b_1 = 1/6$, $\bar{F}_i$ is as defined in equation (A.10), and $\mathbf{x}_{k,l} = x_{1,k,l}, x_{2,k,l}, \ldots, x_{n_x,k,l}$. As with the spatially homogeneous Bonny model, I set $\alpha_i = 1$, $\beta_i = 10^2$, and $\gamma_i = 2$ in smoothing penalties, $s_i^t(\mathbf{x})$ and $s_i^s(\mathbf{x})$, of $r_{\Delta x}(\mathbf{p}, \mathbf{x})$, for all $i \in \{1, 2, \ldots, n_x\}$, to insignificantly modify $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ with a smooth set of state values and to strongly penalize $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ with a jagged set of state values.

3.4.2 Domain Restrictions on Parameters and States

Rate parameters, ω_D, ω_{dD}, ω_E, ω_{ed}, $\omega_{de,m}$, $\omega_{de,c}$, and ω_e, and diffusion coefficients, D_d, D_{de}, and D_e, are only biologically relevant if nonnegative. Thus, I restrict rate parameters and diffusion coefficients to nonnegative values:

$$p \geq 0 \text{ for all } p \in \{\omega_D, \omega_{dD}, \omega_E, \omega_{ed}, \omega_{de,m}, \omega_{de,c}, \omega_e, D_d, D_{de}, D_e\}. \tag{3.7}$$

Parameter $c_{\max}$ dictates the maximum concentration of membrane-bound MinD. Thus, I restrict $c_{\max}$ to values greater than or equal to the maximum MinD data value, $D_{\max}$:

$$c_{\max} \geq D_{\max}. \tag{3.8}$$

For the synthetic spatially-homogeneous data, $D_{\max} = 1.88 \cdot 10^4$ μm^{-2}; for the synthetic traveling-wave data, $D_{\max} = 1.72 \cdot 10^4$ μm^{-2}; and for the synthetic traveling-wave-emergence data, $D_{\max} = 1.88 \cdot 10^4$ μm^{-2}. Concentrations c_d, c_{de}, and c_e are only biologically relevant if nonnegative. Thus, I restrict c_d, c_{de}, and c_e to nonnegative values:

$$c_{i,k} \geq 0 \text{ for all } i \in \{d, de, e\} \text{ and } k \in \mathcal{I}_{\Delta}, \tag{3.9}$$

where $c_{d,k}$, $c_{de,k}$, and $c_{e,k}$ are the values of c_d, c_{de}, and c_e at the k^{th} index of the numerical discretization. Details of overlapping-niche descent on restricted domains are described in Section C.2.3.

3.4.3 Niches

I choose 101 values of λ, λ_k for $k = 1, 2, \ldots, 101$, to define 101 niches. The bounds (B.68) and (B.69), which state that $\breve{r}_{\text{y}}(\lambda) \leq \bar{\varepsilon}$ if $\lambda \leq \bar{\varepsilon}/(1+\bar{\varepsilon})$ and $\breve{r}_{\Delta x}(\lambda) \leq \bar{\varepsilon}$ if $\lambda \geq 1/(1+\bar{\varepsilon})$ for some tolerance $\bar{\varepsilon}$, provide a meaningful guide for the choice of λ_k. Thus, based on the bounds (B.68) and (B.69) with chosen $\bar{\varepsilon} = b^0, b^{-1}, \ldots, b^{-50}$ and base b such that $b^{-50} = 10^{-6}$, I define λ_k for $k = 1, 2, \ldots, 101$ such that

$$\lambda_k = \begin{cases} \dfrac{b^{51-k}}{1+b^{51-k}} & \text{if } k \leq 51 \\ \dfrac{1}{1+b^{51-k}} & \text{if } k > 51. \end{cases} \tag{3.10}$$

My choice of λ_k distributes the values of λ_k for $k = 1, 2, \ldots, 101$ more densely near 0 and 1 and less densely near 0.5. For reference, $\lambda_1 \approx 10^{-6}$, $\lambda_2 \approx 1.3 \cdot 10^{-6}$, $\lambda_{51} = 0.5$, $\lambda_{52} \approx 0.57$, $\lambda_{100} \approx 1 - 1.3 \cdot 10^{-6}$, and $\lambda_{101} \approx 1 - 10^{-6}$.

3.4.4 Calculating Confidence Intervals

I calculate confidence intervals by bootstrapping, given the complex nonlinear relationship between data noise and parameter noise that would not be adequately captured using a (Taylor expansion based) delta method [39]. In doing so, I calculate observable-state residuals,

$$\tilde{\varepsilon}_{j,k} = \text{y}_{j,k} - g_j(\tilde{\mathbf{p}}, \tilde{x}_{1,k}, \ldots, \tilde{x}_{n_x,k}), \tag{3.11}$$

where $\tilde{\mathbf{p}} = \tilde{\mathbf{p}}^{\lambda_{101}}$ and $\tilde{\mathbf{x}} = \tilde{\mathbf{x}}^{\lambda_{101}}$, the parameters and state values that minimize $r(\mathbf{p}, \mathbf{x}; \lambda_{101})$, and $\tilde{x}_{i,k}$ is the value in $\tilde{\mathbf{x}}$ from the i^{th} state and the k^{th} grid index, for $i \in \{1, 2, \ldots, n_x\}$, $j \in \{1, 2, \ldots, n_y\}$, and $k \in \{1, 2, \ldots, n_t\}$. By resampling residuals, I generate $n_b = 10^3$ bootstrap data sets :

$$\text{y}_{j,k} = g_j(\tilde{\mathbf{p}}, \tilde{x}_{1,k}, \ldots, \tilde{x}_{n_x,k}) + \tilde{\varepsilon}_{j,l}, \tag{3.12}$$

where l is randomly sampled with replacement from $\{1, 2, \ldots, n_t\}$, for $j \in \{1, 2, \ldots, n_y\}$ and $k \in \mathcal{I}_\Delta$. I replace observed data values in $r(\mathbf{p}, \mathbf{x}; \lambda)$ with bootstrap data values from the i^{th} bootstrap data set to construct the functional $r_i^b(\mathbf{p}, \mathbf{x}; \lambda)$. Globally minimizing $r_i^b(\mathbf{p}, \mathbf{x}; \lambda)$ using overlapping-niche descent for all $i \in \{1, 2, \ldots, n_b\}$ would be computationally prohibitive. Rather, if residuals are not overly large, the optimal parameters and state values of $r_i^b(\mathbf{p}, \mathbf{x}; \lambda)$ will generally be fairly similar to $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{x}}$. Thus, with $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{x}}$ as initial parameters and state values, I locally optimize $r_i^b(\mathbf{p}, \mathbf{x}; \lambda_b)$ using accelerated descent, for all $i \in \{1, 2, \ldots, n_b\}$, with λ_b chosen

large enough to weight local optimization towards a numerical solution but not so large that $\mathbf{p}$ and $\mathbf{x}$ are fixed near $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{x}}$. Specifically, I choose

$$\lambda_b = \arg\min \left\{ \left| r_{\mathrm{y}}(\tilde{\mathbf{p}}^{\lambda}, \tilde{\mathbf{x}}^{\lambda}) - 10^3 r_{\Delta x}(\tilde{\mathbf{p}}^{\lambda}, \tilde{\mathbf{x}}^{\lambda}) \right| : \lambda \in \{\lambda_1, \lambda_2, \ldots, \lambda_{101}\} \right\}. \tag{3.13}$$

From the n_b local optimizations, I construct a distribution of values for each parameter. From the distribution of values for parameter p_j, I compute the 2.5^{th} and 97.5^{th} percentile values, which I translate into the 95% confidence interval for parameter p_j, for $j \in \{1, 2, \ldots, n_p\}$.

3.5 Fitting Forms of the Bonny Model to Synthetic Data

To ascertain the efficacy of overlapping-niche descent, I fit forms of the Bonny model to synthetic data.

3.5.1 Fitting the Spatially Homogeneous Bonny Model to the Synthetic Spatially-Homogeneous Data

I fit the spatially homogeneous Bonny model (3.2) to the synthetic spatially-homogeneous data using overlapping-niche descent, as described in Section 3.4, on a uniform grid with a grid refinement factor of 1, ${n_\Delta}{n_t}^{-1} = 1$ for n_Δ the number or grid points and n_t the number of data points. I find that $r_{\mathrm{y}}(\tilde{\mathbf{p}}^{\lambda_{101}}, \tilde{\mathbf{x}}^{\lambda_{101}}) = 4.36 \cdot 10^{-4}$, $r_{\Delta x}(\tilde{\mathbf{p}}^{\lambda_{101}}, \tilde{\mathbf{x}}^{\lambda_{101}}) = 1.06 \cdot 10^{-11}$, the mean time per iteration of accelerated descent is $1.46 \cdot 10^{-4}$ s, and the total accelerated descent time is $7.38 \cdot 10^{-1}$ minutes. I calculate the total accelerated descent time as the sum of the maximal accelerated descent time in each generation, as I compute accelerated descent in parallel. Observable-state values of $\tilde{\mathbf{x}}^{\lambda_{101}}$, the state values that minimize $r(\mathbf{p}, \mathbf{x}; \lambda_{101})$, are shown in Figure 3.4.

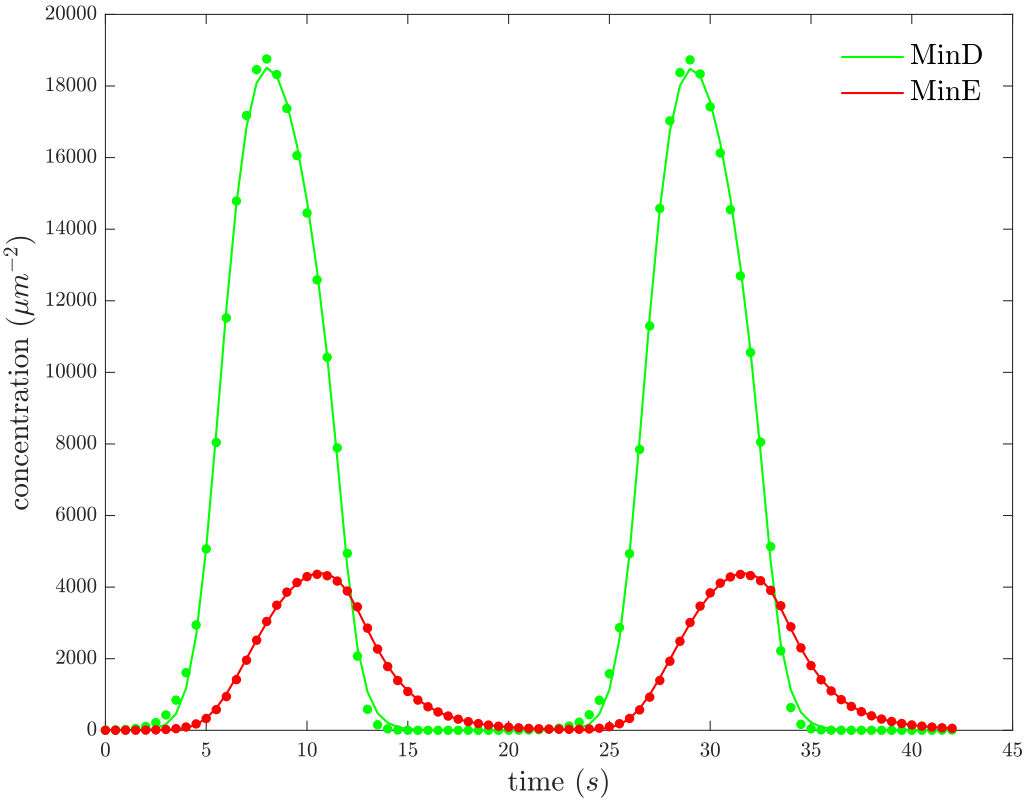


Figure 3.4: The fit of the spatially homogeneous Bonny model to the synthetic spatially-homogeneous data. Observable-state values are shown with solid lines and data values are shown with points. The spatially homogeneous Bonny model fits the synthetic spatially-homogeneous data fairly well. Fitting errors arise from a relatively course numerical discretization.

As is visible in Figure 3.4, the observable-state values of $\tilde{\mathbf{x}}^{\lambda_{101}}$ visibly differ from synthetic data values at some times. This discrepancy stems from a relatively inaccurate method on a relatively coarse grid. On more refined grids, the observable-state values of $\tilde{\mathbf{x}}^{\lambda_{101}}$ fit synthetic data more accurately (shown in Section 3.6) and are visually indistinguishable from the synthetic spatially-homogeneous data (not shown). Parameter estimates from the fit of the spatially homogeneous Bonny model to the synthetic spatially-homogeneous data are shown in Table 3.2.

	true value	estimated value	95% confidence interval	units
$c_{\max}$	$2.75 \cdot 10^4$	$3.02 \cdot 10^4$	$[2.96 \cdot 10^4, 3.09 \cdot 10^4]$	μm^{-2}
ω_D	$5.00 \cdot 10^{-4}$	$0.00 \cdot 10^0$	$[0.00 \cdot 10^0, 2.56 \cdot 10^{-3}]$	$\mu m\ s^{-1}$
ω_E	$1.36 \cdot 10^{-4}$	$1.20 \cdot 10^{-4}$	$[1.17 \cdot 10^{-4}, 1.22 \cdot 10^{-4}]$	$\mu m^3\ s^{-1}$
ω_{dD}	$3.18 \cdot 10^{-3}$	$2.99 \cdot 10^{-3}$	$[2.91 \cdot 10^{-3}, 3.05 \cdot 10^{-3}]$	$\mu m^3\ s^{-1}$
$\omega_{de,c}$	$1.60 \cdot 10^{-1}$	$9.31 \cdot 10^{-2}$	$[8.64 \cdot 10^{-2}, 9.98 \cdot 10^{-2}]$	s^{-1}
$\omega_{de,m}$	$2.52 \cdot 10^0$	$2.59 \cdot 10^0$	$[2.56 \cdot 10^0, 2.62 \cdot 10^0]$	s^{-1}
ω_e	$5.00 \cdot 10^{-1}$	$5.78 \cdot 10^{-1}$	$[5.69 \cdot 10^{-1}, 5.91 \cdot 10^{-1}]$	s^{-1}
ω_{ed}	$4.90 \cdot 10^{-3}$	$4.41 \cdot 10^{-3}$	$[4.35 \cdot 10^{-3}, 4.46 \cdot 10^{-3}]$	$\mu m^2\ s^{-1}$

Table 3.2: Parameter estimates from the fit of the spatially homogeneous Bonny model to the synthetic spatially-homogeneous data. Parameter estimates are generally fairly similar to true parameter values.

3.5.2 Fitting the Traveling Wave Bonny Model to the Synthetic Traveling-Wave Data

I fit the traveling wave Bonny model (3.3) to the synthetic traveling-wave data using overlapping-niche descent, as described in Section 3.4, on a uniform grid with a grid refinement factor of 1, $n_\Delta {n_t}^{-1} = 1$ for n_Δ the number or grid points and n_t the number of data points. I find that the observable-state values of $\tilde{\mathbf{x}}^{\lambda_{101}}$ are visually indistinguishable from the synthetic traveling-wave data shown in Figure 3.2, $r_{\mathrm{y}}(\tilde{\mathbf{p}}^{\lambda_{101}}, \tilde{\mathbf{x}}^{\lambda_{101}}) = 1.61 \cdot 10^{-5}$, $r_{\Delta x}(\tilde{\mathbf{p}}^{\lambda_{101}}, \tilde{\mathbf{x}}^{\lambda_{101}}) = 6.18 \cdot 10^{-17}$, the mean time per iteration of accelerated descent is $1.29 \cdot 10^{-4}$ s, and the total accelerated descent time is $2.69 \cdot 10^{-1}$ minutes. I note that overlapping-niche descent requires a similar amount of time to fit the traveling wave Bonny model to the synthetic traveling-wave data as it does to fit the spatially homogeneous Bonny model to the synthetic spatially homogeneous data ($2.69 \cdot 10^{-1}$ minutes vs. $7.38 \cdot 10^{-1}$ minutes), even though the traveling wave Bonny model is a boundary value problem and the spatially homogeneous Bonny model is an initial value problem. Parameter estimates from the fit of the traveling wave Bonny model to the synthetic traveling-wave data are shown in Table 3.3.

	true value	estimated value	95% confidence interval	units
D_d	$3.00 \cdot 10^{-1}$	$3.37 \cdot 10^{-1}$	$[3.31 \cdot 10^{-1}, 3.46 \cdot 10^{-1}]$	$\mu m^2\ s^{-1}$
D_{de}	$3.00 \cdot 10^{-1}$	$2.42 \cdot 10^{-1}$	$[2.30 \cdot 10^{-1}, 2.53 \cdot 10^{-1}]$	$\mu m^2\ s^{-1}$
D_e	$1.80 \cdot 10^{0}$	$1.68 \cdot 10^{0}$	$[1.65 \cdot 10^{0}, 1.71 \cdot 10^{0}]$	$\mu m^2\ s^{-1}$
$c_{\max}$	$2.75 \cdot 10^{4}$	$2.83 \cdot 10^{4}$	$[2.81 \cdot 10^{4}, 2.84 \cdot 10^{4}]$	μm^{-2}
ω_D	$5.00 \cdot 10^{-4}$	$1.36 \cdot 10^{-3}$	$[5.82 \cdot 10^{-4}, 1.97 \cdot 10^{-3}]$	$\mu m\ s^{-1}$
ω_E	$1.36 \cdot 10^{-4}$	$1.39 \cdot 10^{-4}$	$[1.38 \cdot 10^{-4}, 1.40 \cdot 10^{-4}]$	$\mu m^3\ s^{-1}$
ω_{dD}	$3.18 \cdot 10^{-3}$	$3.07 \cdot 10^{-3}$	$[3.06 \cdot 10^{-3}, 3.08 \cdot 10^{-3}]$	$\mu m^3\ s^{-1}$
$\omega_{de,c}$	$1.60 \cdot 10^{-1}$	$1.72 \cdot 10^{-1}$	$[1.69 \cdot 10^{-1}, 1.76 \cdot 10^{-1}]$	s^{-1}
$\omega_{de,m}$	$2.52 \cdot 10^{0}$	$2.49 \cdot 10^{0}$	$[2.48 \cdot 10^{0}, 2.49 \cdot 10^{0}]$	s^{-1}
ω_e	$5.00 \cdot 10^{-1}$	$4.94 \cdot 10^{-1}$	$[4.87 \cdot 10^{-1}, 4.97 \cdot 10^{-1}]$	s^{-1}
ω_{ed}	$4.90 \cdot 10^{-3}$	$4.84 \cdot 10^{-3}$	$[4.84 \cdot 10^{-3}, 4.86 \cdot 10^{-3}]$	$\mu m^2\ s^{-1}$

Table 3.3: Parameter estimates from the fit of the traveling wave Bonny model to the synthetic traveling-wave data. Parameter estimates are closer to true parameter values than those shown in Table 3.2.

Rate parameter estimates from fitting the traveling wave Bonny model to the synthetic traveling-wave data (shown in Table 3.3) are generally somewhat more accurate with narrower spreads than rate parameter estimates from fitting the spatially homogeneous Bonny model to the synthetic spatially-homogeneous data (shown in Table 3.2). I suspect this occurs because, on average, the traveling wave Bonny model fits the synthetic traveling-wave data somewhat better than the spatially homogeneous Bonny model fits the synthetic spatially-homogeneous data ($r_{\mathrm{y}}(\tilde{\mathbf{p}}^{\lambda_{101}}, \tilde{\mathbf{x}}^{\lambda_{101}}) = 1.61 \cdot 10^{-5}$ vs. $r_{\mathrm{y}}(\tilde{\mathbf{p}}^{\lambda_{101}}, \tilde{\mathbf{x}}^{\lambda_{101}}) = 4.36 \cdot 10^{-4}$).

3.5.3 Fitting the Full Bonny Model to the Synthetic Traveling-Wave-Emergence Data

I fit the full Bonny model (3.1) to the synthetic traveling-wave-emergence data using overlapping-niche descent, as described in Section 3.4, on a uniform grid with a grid refinement factor of 1, $n_{\Delta_t} n_{\Delta_s} {n_t}^{-1} {n_s}^{-1} = 1$ for n_{Δ_t} and n_{Δ_s} the number or temporal and spatial grid points and n_t and n_s the number of temporal and spatial data points. I find that the observable-state values of $\tilde{\mathbf{x}}^{\lambda_{101}}$ are visually indistinguishable from the synthetic traveling-wave-emergence data shown in Figure 3.3, $r_{\mathrm{y}}(\tilde{\mathbf{p}}^{\lambda_{101}}, \tilde{\mathbf{x}}^{\lambda_{101}}) = 1.29 \cdot 10^{-6}$, $r_{\Delta x}(\tilde{\mathbf{p}}^{\lambda_{101}}, \tilde{\mathbf{x}}^{\lambda_{101}}) = 1.98 \cdot 10^{-16}$, the mean time per iteration of accelerated descent is $6.03 \cdot 10^{-3}$ s, and the total accelerated descent time is 48.0 minutes. Parameter estimates from the fit of the full Bonny model to the synthetic traveling-wave-emergence data are shown in Table 3.4.

	true value	estimated value	95% confidence interval	units
D_d	$3.00 \cdot 10^{-1}$	$2.98 \cdot 10^{-1}$	$[2.98 \cdot 10^{-1}, 2.99 \cdot 10^{-1}]$	$\mu m^2\ s^{-1}$
D_{de}	$3.00 \cdot 10^{-1}$	$3.04 \cdot 10^{-1}$	$[3.03 \cdot 10^{-1}, 3.05 \cdot 10^{-1}]$	$\mu m^2\ s^{-1}$
D_e	$1.80 \cdot 10^{0}$	$1.83 \cdot 10^{0}$	$[1.83 \cdot 10^{0}, 1.83 \cdot 10^{0}]$	$\mu m^2\ s^{-1}$
$c_{\max}$	$2.75 \cdot 10^{4}$	$2.75 \cdot 10^{4}$	$[2.75 \cdot 10^{4}, 2.75 \cdot 10^{4}]$	μm^{-2}
ω_D	$5.00 \cdot 10^{-4}$	$0.00 \cdot 10^{0}$	$[0.00 \cdot 10^{0}, 1.83 \cdot 10^{-5}]$	$\mu m\ s^{-1}$
ω_E	$1.36 \cdot 10^{-4}$	$1.36 \cdot 10^{-4}$	$[1.35 \cdot 10^{-4}, 1.36 \cdot 10^{-4}]$	$\mu m^3\ s^{-1}$
ω_{dD}	$3.18 \cdot 10^{-3}$	$3.18 \cdot 10^{-3}$	$[3.18 \cdot 10^{-3}, 3.18 \cdot 10^{-3}]$	$\mu m^3\ s^{-1}$
$\omega_{de,c}$	$1.60 \cdot 10^{-1}$	$1.59 \cdot 10^{-1}$	$[1.58 \cdot 10^{-1}, 1.59 \cdot 10^{-1}]$	s^{-1}
$\omega_{de,m}$	$2.52 \cdot 10^{0}$	$2.52 \cdot 10^{0}$	$[2.52 \cdot 10^{0}, 2.52 \cdot 10^{0}]$	s^{-1}
ω_e	$5.00 \cdot 10^{-1}$	$5.00 \cdot 10^{-1}$	$[5.00 \cdot 10^{-1}, 5.01 \cdot 10^{-1}]$	s^{-1}
ω_{ed}	$4.90 \cdot 10^{-3}$	$4.85 \cdot 10^{-3}$	$[4.85 \cdot 10^{-3}, 4.85 \cdot 10^{-3}]$	$\mu m^2\ s^{-1}$

Table 3.4: Parameter estimates from the fit of the full Bonny model to the synthetic traveling-wave-emergence data. Parameter estimates are very similar to true parameter values except for the parameter ω_D, which seems to play a small role is the overall dynamics of the Bonny model.

Parameter estimates from fitting the full Bonny model to the synthetic traveling-wave-emergence data (shown in Figure 3.4) are generally somewhat more accurate with narrower spreads than parameter estimates from fitting the traveling wave Bonny model to the synthetic traveling-wave data (shown in Table 3.3). I suspect this occurs because, on average, the full Bonny model fits the synthetic traveling-wave-emergence data somewhat better than the traveling wave Bonny model fits the synthetic traveling-wave data ($r_{\text{y}}(\tilde{\mathbf{p}}^{\lambda_{101}}, \tilde{\mathbf{x}}^{\lambda_{101}}) = 1.29 \cdot 10^{-6}$ vs. $r_{\text{y}}(\tilde{\mathbf{p}}^{\lambda_{101}}, \tilde{\mathbf{x}}^{\lambda_{101}}) = 1.61 \cdot 10^{-5}$).

Neither the spatially homogeneous Bonny model, the traveling wave Bonny model, nor the full Bonny model accurately estimate the nonzero value ω_D when fitting respective synthetic data (see Tables 3.2, 3.3, and 3.4). Yet, the spatially homogeneous Bonny model (on grids more refined than when $n_\Delta {n_t}^{-1} = 1$), the traveling wave Bonny model, and the full Bonny model very accurately fit respective synthetic data. Thus, it appears that, beyond allowing an initial increase in c_d from a homogeneous initial condition, ω_D plays very little role in the overall dynamics of the Bonny systems.

As shown for the spatially homogeneous Bonny model in Table 3.2, the traveling wave Bonny model in Table 3.3, and the full Bonny model in Table 3.4, 95% confidence intervals often do not include true parameter values. I suspect this occurs because errors in fitting arise from discretization errors in the numerical methods. As such, residuals are small and are not independent or identically distributed. Thus, when calculating confidence intervals by bootstrapping as described in Section 3.4.4, bootstrap data sets do not significantly differ from synthetic data and errors in bootstrap data sets do not accurately represent discretization errors in the numerical methods. Thus, 95% confidence intervals are often fairly narrow and may not include true parameter values.

3.6 Comparing Overlapping-Niche Descent to a Numerical-Integration-Based Method

I compare the performance of overlapping-niche descent to the performance of a numerical-integration-based parameter optimization method. For a balanced comparison, I construct a variant of overlapping-niche descent that omits $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ from the objective function and instead solves the differential equation numerically at each step. It also uses a single niche, so I refer to it as single-niche solution descent (SNSD). SNSD optimizes over parameters and initial conditions to minimize $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$, with numerical solution values $\mathbf{x}$. Details of SNSD are described in Section E.2.

To highlight scenarios in which overlapping-niche descent outperforms SNSD, I compare the performance of overlapping-niche descent to the performance of SNSD on a set of differential equations systems that vary only in the size of the system. I construct the Bonny $\times\, n$ model, a differential equation system consisting of n independent copies of the spatially homogeneous Bonny model (3.2), with $3n$ states, $8n$ parameters, and $2n$ observable states. Accordingly, I generate synthetic spatially-homogeneous $\times\, n$ data using n copies of the synthetic spatially-homogeneous data. I fit the Bonny $\times\, n$ model to the synthetic spatially-homogeneous $\times\, n$ data, for $n = 1, 2, \ldots, 5$, using overlapping-niche descent, as described in Section 3.4, and using SNSD. For both, I use the backward Euler scheme on a set of uniform grids with grid refinement factors of 1, 2, 3, and 4, $n_{\Delta} {n_t}^{-1} = 1, 2, 3, 4$ for n_{Δ} the number or grid points and n_t the number of data points. I construct $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$, $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x})$, and $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ for the Bonny $\times\, n$ model as described in Section 3.4.1 for the spatially homogeneous Bonny model. $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ is normalized by the number of observable states. Thus, for $\check{\mathbf{p}}$ and $\check{\mathbf{x}}$, the parameter and numerical solution values that minimize $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$, the value of $r_{\mathrm{y}}(\check{\mathbf{p}}, \check{\mathbf{x}})$ is identical for the Bonny $\times\, n$ model with all $n \in \mathbb{N}^{+}$. In SNSD, I calculate numerical solutions using the backward Euler method, solve nonlinear systems using Newton's method with an absolute termination tolerance of 10^{-3}, and solve matrix equations using Gaussian elimination. Ultimately, I calculate $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{x}}$, approximations of $\check{\mathbf{p}}$ and $\check{\mathbf{x}}$. For overlapping-niche descent, $\tilde{\mathbf{p}} = \tilde{\mathbf{p}}^{\lambda_{101}}$ and $\tilde{\mathbf{x}} = \tilde{\mathbf{x}}^{\lambda_{101}}$, the parameters and state values that minimize $r(\mathbf{p}, \mathbf{x}; \lambda_{101})$. Results are shown in Tables 3.5, 3.6, 3.7, and 3.8.

overlapping-niche descent				
	$n_\Delta = n_t$	$n_\Delta = 2n_t$	$n_\Delta = 3n_t$	$n_\Delta = 4n_t$
Bonny × 1	$4.36 \cdot 10^{-4}$	$1.14 \cdot 10^{-4}$	$5.18 \cdot 10^{-5}$	$2.95 \cdot 10^{-5}$
Bonny × 2	$4.38 \cdot 10^{-4}$	$1.14 \cdot 10^{-4}$	$5.18 \cdot 10^{-5}$	$2.95 \cdot 10^{-5}$
Bonny × 3	$4.36 \cdot 10^{-4}$	$1.14 \cdot 10^{-4}$	$5.18 \cdot 10^{-5}$	$2.95 \cdot 10^{-5}$
Bonny × 4	$4.35 \cdot 10^{-4}$	$1.14 \cdot 10^{-4}$	$5.19 \cdot 10^{-5}$	$2.95 \cdot 10^{-5}$
Bonny × 5	$4.31 \cdot 10^{-4}$	$1.14 \cdot 10^{-4}$	$5.19 \cdot 10^{-5}$	$2.95 \cdot 10^{-5}$

SNSD				
	$n_\Delta = n_t$	$n_\Delta = 2n_t$	$n_\Delta = 3n_t$	$n_\Delta = 4n_t$
Bonny × 1	$4.53 \cdot 10^{-4}$	$1.14 \cdot 10^{-4}$	$5.18 \cdot 10^{-5}$	$2.95 \cdot 10^{-5}$
Bonny × 2	$4.53 \cdot 10^{-4}$	$1.14 \cdot 10^{-4}$	$5.18 \cdot 10^{-5}$	$2.95 \cdot 10^{-5}$
Bonny × 3	$4.53 \cdot 10^{-4}$	$1.14 \cdot 10^{-4}$	$5.18 \cdot 10^{-5}$	$2.95 \cdot 10^{-5}$
Bonny × 4	$4.53 \cdot 10^{-4}$	$1.14 \cdot 10^{-4}$	$5.18 \cdot 10^{-5}$	$2.95 \cdot 10^{-5}$
Bonny × 5	$\mathbf{1.37 \cdot 10^{-1}}$	$1.14 \cdot 10^{-4}$	$\mathbf{1.37 \cdot 10^{-1}}$	$\mathbf{1.37 \cdot 10^{-1}}$

Table 3.5: Values of $r_{\mathrm{y}}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from overlapping-niche descent and SNSD. $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{x}}$ approximate $\check{\mathbf{p}}$ and $\check{\mathbf{x}}$, the parameters and state values of the optimal data-fitting numerical solution, for the fit of the Bonny × n model, a differential equation system consisting of n independent copies of the spatially homogeneous Bonny model (3.2), to the synthetic spatially-homogeneous × n data, which consists of n copies of the synthetic spatially-homogeneous data, for $n \in \{1, 2, \ldots, 5\}$. Bold values are shown for emphasis. Values of $r_{\mathrm{y}}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from overlapping-niche descent and SNSD are similar except for $n = 5$ in the Bonny × n model when SNSD fails to find the optimal data-fitting numerical solution.

overlapping-niche descent				
	$n_\Delta = n_t$	$n_\Delta = 2n_t$	$n_\Delta = 3n_t$	$n_\Delta = 4n_t$
Bonny × 1	$1.06 \cdot 10^{-11}$	$4.82 \cdot 10^{-17}$	$4.69 \cdot 10^{-17}$	$4.61 \cdot 10^{-16}$
Bonny × 2	$1.50 \cdot 10^{-11}$	$7.63 \cdot 10^{-16}$	$1.54 \cdot 10^{-17}$	$9.60 \cdot 10^{-18}$
Bonny × 3	$1.31 \cdot 10^{-11}$	$4.40 \cdot 10^{-16}$	$8.92 \cdot 10^{-18}$	$1.66 \cdot 10^{-17}$
Bonny × 4	$2.31 \cdot 10^{-11}$	$2.78 \cdot 10^{-17}$	$2.22 \cdot 10^{-15}$	$1.39 \cdot 10^{-15}$
Bonny × 5	$5.04 \cdot 10^{-11}$	$2.77 \cdot 10^{-17}$	$8.10 \cdot 10^{-17}$	$2.89 \cdot 10^{-17}$

SNSD				
	$n_\Delta = n_t$	$n_\Delta = 2n_t$	$n_\Delta = 3n_t$	$n_\Delta = 4n_t$
Bonny × 1	$1.42 \cdot 10^{-12}$	$1.75 \cdot 10^{-12}$	$2.03 \cdot 10^{-12}$	$1.78 \cdot 10^{-12}$
Bonny × 2	$1.42 \cdot 10^{-12}$	$1.75 \cdot 10^{-12}$	$2.03 \cdot 10^{-12}$	$1.78 \cdot 10^{-12}$
Bonny × 3	$1.42 \cdot 10^{-12}$	$1.75 \cdot 10^{-12}$	$2.03 \cdot 10^{-12}$	$1.78 \cdot 10^{-12}$
Bonny × 4	$1.42 \cdot 10^{-12}$	$1.75 \cdot 10^{-12}$	$2.03 \cdot 10^{-12}$	$1.78 \cdot 10^{-12}$
Bonny × 5	$1.14 \cdot 10^{-12}$	$1.75 \cdot 10^{-12}$	$1.62 \cdot 10^{-12}$	$1.42 \cdot 10^{-12}$

Table 3.6: Values of $r_{\Delta x}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from overlapping-niche descent and SNSD. Notation is as defined in Table 3.5. Values of $r_{\Delta x}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from SNSD are nonzero because numerical solution values are calculated implicitly using Newton's method in SNSD. Values of $r_{\Delta x}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from overlapping-niche descent are significantly less than values of $r_{\Delta x}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from SNSD except on unrefined grids, where $n_\Delta = n_t$.

As shown in Table 3.5, $r_{\mathrm{y}}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from overlapping-niche descent is essentially equal to $r_{\mathrm{y}}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from SNSD on each grid with $n_\Delta = 2n_t, 3n_t, 4n_t$ and for each Bonny $\times$ n model with $n = 1, 2, 3, 4$. $r_{\mathrm{y}}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from overlapping-niche descent is slightly less than $r_{\mathrm{y}}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from SNSD on the grid with $n_\Delta = n_t$ and for each Bonny $\times$ n model with $n = 1, 2, 3, 4$. However, each integral representation of $r_{\mathrm{y}}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from overlapping-niche descent, as defined in equation (2.26b), is similar to $r_{\mathrm{y}}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from SNSD (not shown), and each $r_{\Delta x}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from overlapping-niche descent is larger than $r_{\Delta x}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from SNSD, as shown in Table 3.6. Thus, I expect larger λ in overlapping-niche descent to decrease the value of $r_{\Delta x}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ and increase the value of $r_{\mathrm{y}}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ to be commensurate with $r_{\mathrm{y}}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from SNSD. $r_{\mathrm{y}}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from overlapping-niche descent are essentially equal on each grid with $n_\Delta = 2n_t, 3n_t, 4n_t$ and for each Bonny $\times$ n model with $n = 1, 2, 3, 4, 5$. Somewhat similarly, $r_{\mathrm{y}}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from SNSD are essentially equal on each grid with $n_\Delta = n_t, 2n_t, 3n_t, 4n_t$ and for each Bonny $\times$ n model with $n = 1, 2, 3, 4$. However, $r_{\mathrm{y}}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from SNSD is significantly larger for the Bonny $\times$ n model with $n = 5$ than for $n = 1, 2, 3, 4$ on each grid with $n_\Delta = n_t$, $n_\Delta = 3n_t$, and $n_\Delta = 4n_t$. Also, as shown in Table 3.6, $r_{\Delta x}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from overlapping-niche descent is no greater than $r_{\Delta x}(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ from SNSD on each grid with $n_\Delta = 2n_t, 3n_t, 4n_t$ and for each Bonny $\times$ n model with $n = 1, 2, 3, 4, 5$. Therefore, collectively, overlapping-niche descent appears to find $\check{\mathbf{p}}$ and $\check{\mathbf{x}}$, the parameter and numerical solution values that minimize $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$, on each grid with $n_\Delta = n_t, 2n_t, 3n_t, 4n_t$ and for each Bonny $\times$ n model with $n = 1, 2, 3, 4, 5$. Whereas, SNSD fails to find $\check{\mathbf{p}}$ and $\check{\mathbf{x}}$ on each grid with $n_\Delta = n_t$, $n_\Delta = 3n_t$, and $n_\Delta = 4n_t$ for the Bonny $\times$ n model with $n = 5$. Ultimately, overlapping-niche descent appears to find the optimal data-fitting numerical solution more robustly than SNSD.

A system of differential equations often admits a variety of parameter dependent solution behaviors. Bifurcations separate the solution space, and thus the numerical solution space, into regions with qualitatively different behaviors. In numerical-integration-based methods, including SNSD, parameters and initial conditions entirely define numerical solutions. Thus, a numerical-integration-based method can only find the optimal data-fitting numerical solution if optimization begins with a set of parameters and initial conditions of a numerical solution with the same qualitative behavior as the optimal data-fitting numerical solution.

As a system of differential equations increases in complexity, the system admits a greater variety of solution behaviors with more bifurcations, and the likelihood of randomly finding a set of parameters and initial conditions of a numerical solution with the same qualitative behavior as the optimal data-fitting numerical solution decreases. Also, as the number of parameters increases in a system of differential equations, simply by dimensional scaling, the likelihood of randomly finding a set of parameters and initial conditions of a numerical solution with the same qualitative behavior as the optimal data-fitting numerical solution decreases. Thus, in SNSD, as n increases in the Bonny $\times$ n model, the likelihood of finding the parameters and initial conditions of the optimal data-fitting numerical solution decreases.

In overlapping-niche descent, state values, beyond those of initial conditions, directly guide optimization. Thus, even if random parameters and initial conditions are not those of a numerical

solution with the same qualitative behavior as the optimal data-fitting numerical solution, state values orient optimization towards the parameters and state values of a numerical solution with the same qualitative behavior as data, positioning the optimization routine to potentially find the parameters and state values of the optimal data-fitting numerical solution. This, I believe, is why overlapping-niche descent is more robust than SNSD for the Bonny $\times$ n model with $n = 5$. Accordingly, I expect overlapping-niche descent to be more robust than other numerical-integration-based methods for the Bonny $\times$ n model with $n = 5$, and for complex models in general.

overlapping-niche descent				
	$n_\Delta = n_t$	$n_\Delta = 2n_t$	$n_\Delta = 3n_t$	$n_\Delta = 4n_t$
Bonny $\times$ 1	$1.46 \cdot 10^{-4}$	$2.87 \cdot 10^{-4}$	$4.54 \cdot 10^{-4}$	$5.68 \cdot 10^{-4}$
Bonny $\times$ 2	$4.18 \cdot 10^{-4}$	$8.23 \cdot 10^{-4}$	$1.26 \cdot 10^{-3}$	$1.63 \cdot 10^{-4}$
Bonny $\times$ 3	$9.41 \cdot 10^{-4}$	$1.84 \cdot 10^{-3}$	$2.80 \cdot 10^{-3}$	$3.73 \cdot 10^{-3}$
Bonny $\times$ 4	$1.65 \cdot 10^{-3}$	$3.30 \cdot 10^{-3}$	$5.03 \cdot 10^{-3}$	$6.58 \cdot 10^{-3}$
Bonny $\times$ 5	$2.70 \cdot 10^{-3}$	$5.23 \cdot 10^{-3}$	$7.99 \cdot 10^{-3}$	$1.05 \cdot 10^{-2}$

SNSD				
	$n_\Delta = n_t$	$n_\Delta = 2n_t$	$n_\Delta = 3n_t$	$n_\Delta = 4n_t$
Bonny $\times$ 1	$1.39 \cdot 10^{-3}$	$1.16 \cdot 10^{-3}$	$1.80 \cdot 10^{-3}$	$2.50 \cdot 10^{-3}$
Bonny $\times$ 2	$1.04 \cdot 10^{-2}$	$2.09 \cdot 10^{-2}$	$2.25 \cdot 10^{-2}$	$3.03 \cdot 10^{-2}$
Bonny $\times$ 3	$3.98 \cdot 10^{-2}$	$7.85 \cdot 10^{-2}$	$9.47 \cdot 10^{-2}$	$1.54 \cdot 10^{-1}$
Bonny $\times$ 4	$9.72 \cdot 10^{-2}$	$2.13 \cdot 10^{-1}$	$3.50 \cdot 10^{-1}$	$4.48 \cdot 10^{-1}$
Bonny $\times$ 5	$2.38 \cdot 10^{-1}$	$5.00 \cdot 10^{-1}$	$9.44 \cdot 10^{-1}$	1.26

Table 3.7: Mean time per iteration of descent from overlapping-niche descent and SNSD. Notation is as defined in Table 3.5. Times are shown in seconds. SNSD requires more time for an iteration of descent than overlapping-niche descent, and the difference in required time increases as the system of differential equations increases in size.

overlapping-niche descent				
	$n_\Delta = n_t$	$n_\Delta = 2n_t$	$n_\Delta = 3n_t$	$n_\Delta = 4n_t$
Bonny $\times$ 1	$7.38 \cdot 10^{-1}$	$4.35 \cdot 10^{-1}$	$4.86 \cdot 10^{-1}$	$6.26 \cdot 10^{-1}$
Bonny $\times$ 2	1.23	1.34	3.24	3.68
Bonny $\times$ 3	$1.03 \cdot 10^{1}$	2.90	8.58	7.25
Bonny $\times$ 4	7.55	8.69	$1.32 \cdot 10^{1}$	$1.40 \cdot 10^{1}$
Bonny $\times$ 5	5.37	$1.34 \cdot 10^{1}$	$2.47 \cdot 10^{1}$	$2.09 \cdot 10^{1}$

SNSD				
	$n_\Delta = n_t$	$n_\Delta = 2n_t$	$n_\Delta = 3n_t$	$n_\Delta = 4n_t$
Bonny $\times$ 1	2.25	1.72	2.22	4.80
Bonny $\times$ 2	$2.56 \cdot 10^{1}$	$5.34 \cdot 10^{1}$	$6.27 \cdot 10^{1}$	$5.70 \cdot 10^{1}$
Bonny $\times$ 3	$1.41 \cdot 10^{2}$	$2.45 \cdot 10^{2}$	$3.61 \cdot 10^{2}$	$4.33 \cdot 10^{2}$
Bonny $\times$ 4	$5.32 \cdot 10^{2}$	$8.87 \cdot 10^{2}$	$1.24 \cdot 10^{3}$	$2.99 \cdot 10^{3}$
Bonny $\times$ 5	$1.20 \cdot 10^{3}$	$2.50 \cdot 10^{3}$	$3.15 \cdot 10^{3}$	$3.86 \cdot 10^{3}$

Table 3.8: Total descent time from overlapping-niche descent and SNSD. Notation is as defined in Table 3.5. Times are shown in minutes. The total accelerated descent time is the sum of the maximal accelerated descent time in each generation, as accelerated descent is calculated in parallel. SNSD requires more time for descent than overlapping-niche descent, and the difference in required time increases as the system of differential equations increases in size.

As shown in Tables 3.7 and 3.8 for the Bonny $\times$ 1 model, computational times, the mean time per iteration of descent and the total descent time, of SNSD range from about 3 to about 10 times larger than corresponding computational times of overlapping-niche descent. As n increases in the Bonny $\times$ n model, differences in computational times between SNSD and overlapping-niche descent increase. Notably, for the Bonny $\times$ 5 model, computational times of SNSD range from about 90 to about 220 times larger than corresponding computational times of overlapping-niche descent. I explore this result in Section D.1, where I count the computational complexity of overlapping niche descent and a variety of numerical-integration-based methods, including SNSD. Under relatively general assumptions, I find that overlapping-niche descent outperforms the numerical-integration-based methods and the difference in performance increases with increasing system size, especially with implicit methods and with partial differential equations.

3.7 Noisy Data and Incomplete Modeling

Real data is often noisy and models of real data are often incomplete. As such, I explore differences in the character of fits for a complete model with noisy data and an incomplete model with noiseless data. In doing so, I show how parameters and state values from overlapping-niche descent as $\lambda \to 0^+$ can inform model shortcomings and potential model improvements. I also provide an example that shows how a parameter from overlapping-niche descent differs over $\lambda \in (0, 1)$ for a fit of a complete model to noisy data and a fit of an incomplete model to noiseless data.

To explore differences in the character of fits for a complete model with noisy data and an incomplete model with noiseless data, I generate noisy data and an incomplete model. I can easily add noise to synthetic data to generate noisy data. For an instructive example of an incomplete model, I seek an incomplete variant of one of the forms of the Bonny model that visibly, but not overly, alters the fit to synthetic data. Removing a reaction term from the spatially homogeneous Bonny model leads to either no visible difference or a dramatic difference in the fit to the synthetic spatially homogeneous data (not shown). Thus, I do not construct an incomplete model by removing a reaction term from one of the forms of the Bonny model. Instead, as the synthetic spatially homogeneous data and the synthetic traveling-wave data are visibly different but not so dissimilar in shape, I generate an incomplete traveling wave model by imposing zero diffusion in the traveling wave Bonny model (3.3), $D_d = 0\ \mu m^2\ s^{-1}$, $D_{de} = 0\ \mu m^2\ s^{-1}$, and $D_e = 0\ \mu m^2\ s^{-1}$. For corresponding noisy data, I generate noisy traveling-wave data by adding random errors to the synthetic traveling-wave data. For noisy traveling-wave MinD data, I distribute random errors normally with a mean of zero and a standard deviation that is 0.05 times the range of synthetic traveling-wave MinD data. For noisy traveling-wave MinE data, I distribute random errors normally with a mean of zero and a standard deviation that is 0.05 times the range of synthetic traveling-wave MinE data. In doing so, I restrict noisy traveling-wave data to non-negative values for physical relevance.

Using overlapping-niche descent, as described in Section 3.4, on a uniform grid with a grid refinement factor of 1, ${n_\Delta}{n_t}^{-1} = 1$ for n_Δ the number or grid points and n_t the number of data points, I fit the traveling wave Bonny model to the noisy traveling-wave data, and I fit the incomplete traveling wave model to the synthetic traveling-wave data. Respectively, I find that $r_{\mathrm{y}}(\tilde{\mathbf{p}}^{\lambda_{101}}, \tilde{\mathbf{x}}^{\lambda_{101}}) = 7.30 \cdot 10^{-3}$ and $r_{\mathrm{y}}(\tilde{\mathbf{p}}^{\lambda_{101}}, \tilde{\mathbf{x}}^{\lambda_{101}}) = 3.13 \cdot 10^{-3}$, $r_{\Delta x}(\tilde{\mathbf{p}}^{\lambda_{101}}, \tilde{\mathbf{x}}^{\lambda_{101}}) = 3.40 \cdot 10^{-15}$ and $r_{\Delta x}(\tilde{\mathbf{p}}^{\lambda_{101}}, \tilde{\mathbf{x}}^{\lambda_{101}}) = 5.84 \cdot 10^{-15}$, the mean times per iteration of accelerated descent are $1.28 \cdot 10^{-4}\ s$ and $1.26 \cdot 10^{-4}\ s$, and the total accelerated descent times are $3.27 \cdot 10^{-1}$ minutes and $2.70 \cdot 10^{-1}$ minutes. Observable-state values of $\tilde{\mathbf{x}}^{\lambda_{101}}$ are shown in Figure 3.5.

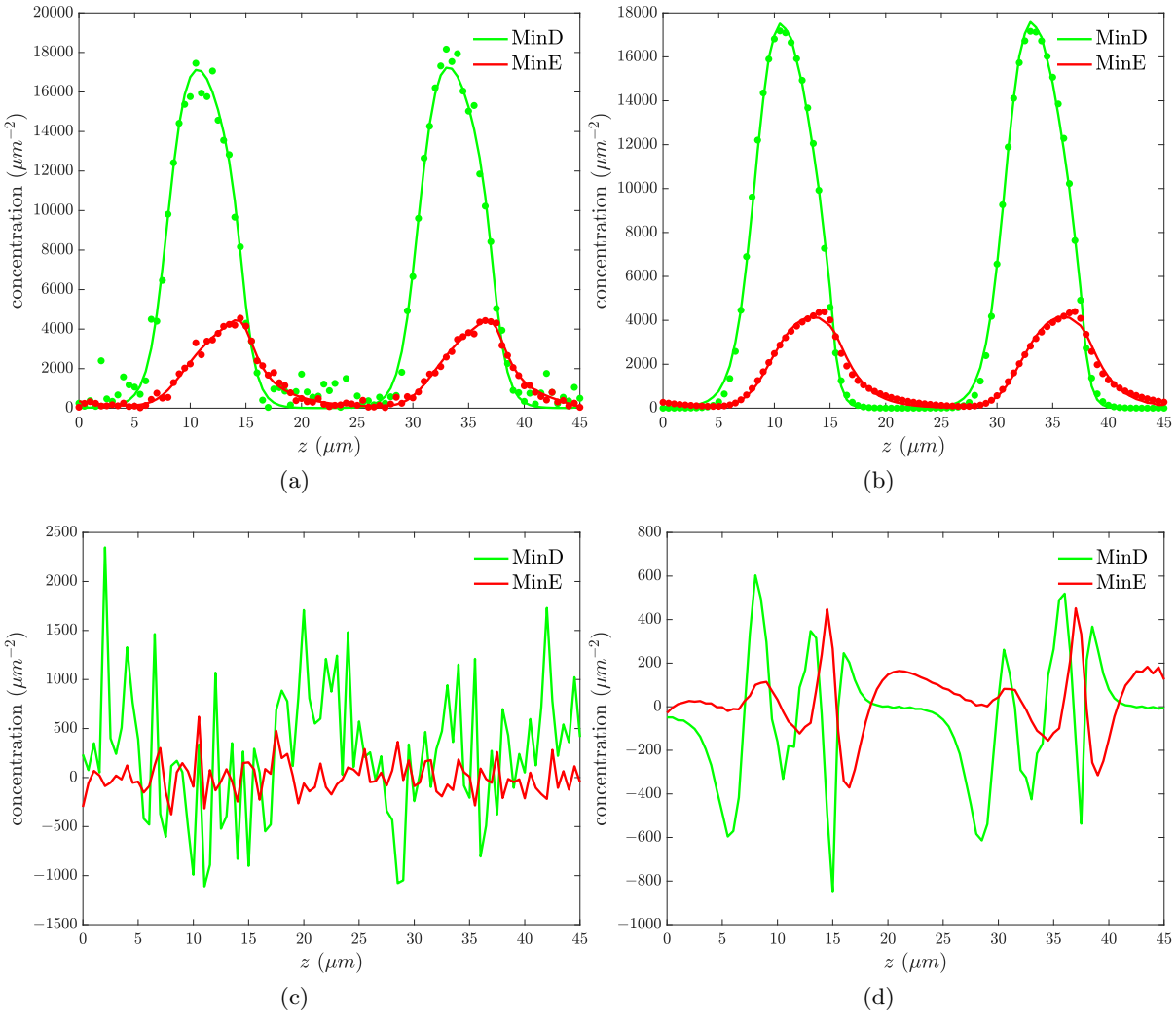


Figure 3.5: Observable-state errors for noisy-data and incomplete-model fits. For the fit of the traveling wave Bonny model to the noisy traveling-wave data, the observable-state values of $\tilde{\mathbf{x}}^{\lambda_{101}}$ and data values are shown in (a) and differences in the observable-state values of $\tilde{\mathbf{x}}^{\lambda_{101}}$ and data values are shown in (c). For the the fit of the incomplete traveling wave model to the synthetic traveling-wave data, the observable-state values of $\tilde{\mathbf{x}}^{\lambda_{101}}$ and data values are shown in (b) and differences in the observable-state values of $\tilde{\mathbf{x}}^{\lambda_{101}}$ and data values are shown in (d). In (a) and (b), observable-state values are shown with solid lines and data values are shown with points. Observable-state errors appear to be uncorrelated in z for the noisy-data fit and highly correlated in z for the incomplete-model fit.

As is visible in Figure 3.5, for the traveling wave Bonny model fit to the noisy traveling-wave data, differences in the observable-state values of $\tilde{\mathbf{x}}^{\lambda_{101}}$ and data values appear to be mostly uncorrelated in z (with a Durbin-Watson statistic in MinD of 1.39 and a Durbin-Watson statistic in MinE of 2.12, where a Durbin-Watson statistic of 2 indicates no autocorrelation in residuals and a Durbin-Watson statistic closer to 0 indicates a greater positive autocorrelation in residuals), reflecting random error; whereas, for the incomplete traveling wave model fit to the synthetic

traveling-wave data, differences in the observable-state values of $\tilde{\mathbf{x}}^{\lambda_{101}}$ and data values appear to be highly correlated in z (with a Durbin-Watson statistic in MinD of 0.52 and a Durbin-Watson statistic in MinE of 0.38), reflecting modeling error. Differences in the observable-state values of $\tilde{\mathbf{x}}^{\lambda_{101}}$ and data values may indicate the existence of modeling error, but they provide little insight into potential sources of the modeling error. Alternatively, I consider numerical discretizations involving $\tilde{\mathbf{p}}^{\lambda_1}$ and $\tilde{\mathbf{x}}^{\lambda_1}$, the parameters and state values closest to a numerical solution given that observable state values (very nearly) match data. Differences in $\Delta^1 x_{i,k}$ and $F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$, as defined in equation (3.5), for $\mathbf{p} = \tilde{\mathbf{p}}^{\lambda_1}$ and $\mathbf{x} = \tilde{\mathbf{x}}^{\lambda_1}$ are the minimal differences in $\Delta^1 x_{i,k}$ and $F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ imposed by data, as measured by $r_{\Delta x}(\mathbf{p}, \mathbf{x})$, for all $i \in \{1, 2, \ldots, n_x\}$ and all k in $\mathcal{I}_\Delta$. Thus, differences in $\Delta^1 x_{i,k}$ and $F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ for $\mathbf{p} = \tilde{\mathbf{p}}^{\lambda_1}$ and $\mathbf{x} = \tilde{\mathbf{x}}^{\lambda_1}$ may reveal model shortcomings and point to changes in a model that can be made to bring the model closer to data. I plot differences in $\Delta^1 x_{i,k}$ and $F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ for $\mathbf{p} = \tilde{\mathbf{p}}^{\lambda_1}$ and $\mathbf{x} = \tilde{\mathbf{x}}^{\lambda_1}$ in Figure 3.6 for the fit of the traveling wave Bonny model to the noisy traveling-wave data and for the fit of the incomplete traveling wave model to the synthetic traveling-wave data, for the state $x_i = c_{de}$.

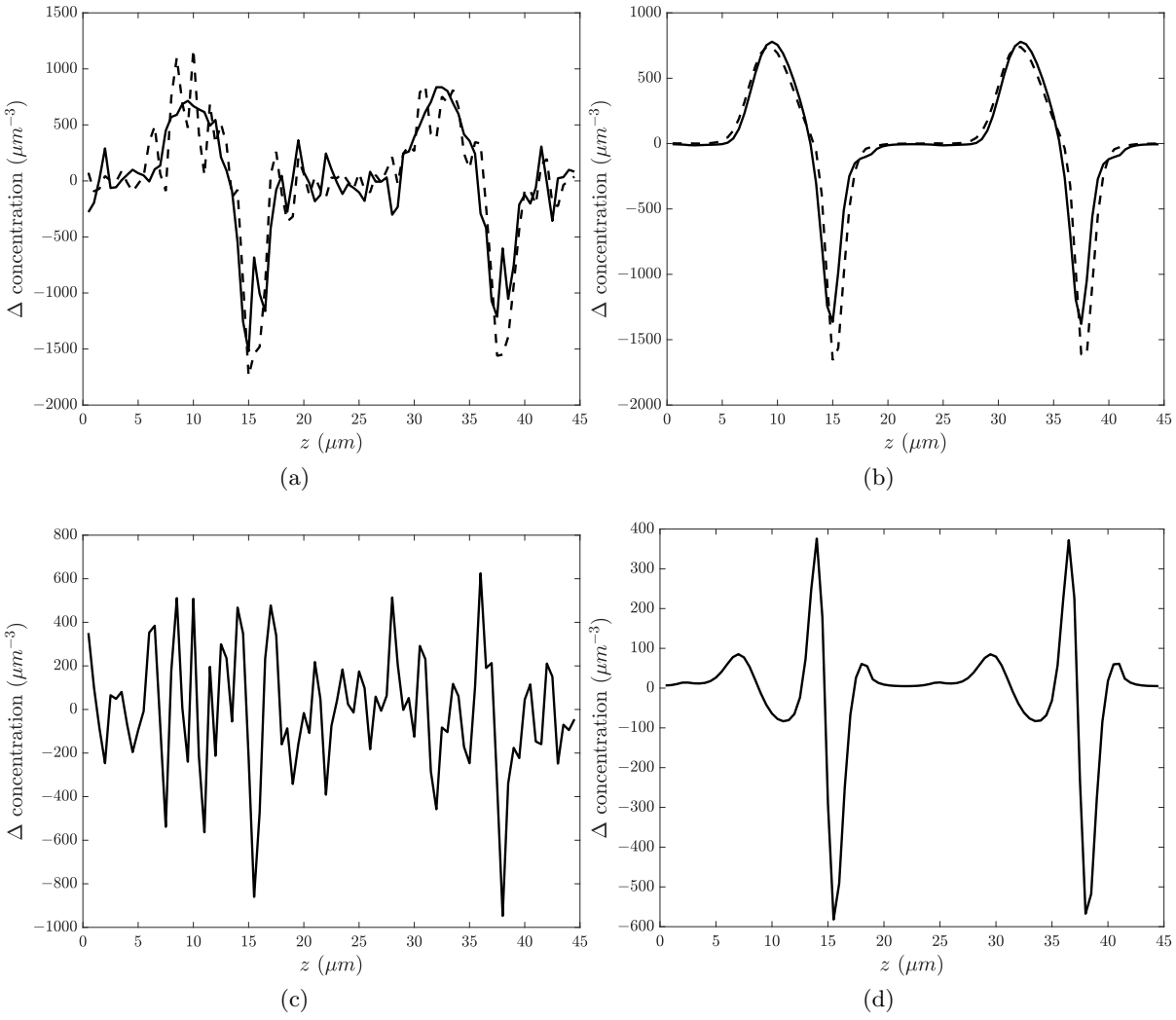


Figure 3.6: Numerical solution errors for noisy-data and incomplete-model fits. Values are shown for the state $x_i = c_{de}$. For the fit of the traveling wave Bonny model to the noisy traveling-wave data, $\Delta^1 x_{i,k}$ and $F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ for $\mathbf{p} = \tilde{\mathbf{p}}^{\lambda_1}$ and $\mathbf{x} = \tilde{\mathbf{x}}^{\lambda_1}$ are shown in (a) and $\Delta^1 x_{i,k} - F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ for $\mathbf{p} = \tilde{\mathbf{p}}^{\lambda_1}$ and $\mathbf{x} = \tilde{\mathbf{x}}^{\lambda_1}$ is shown in (c). For the fit of the incomplete traveling wave model to the synthetic traveling-wave data, $\Delta^1 x_{i,k}$ and $F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ for $\mathbf{p} = \tilde{\mathbf{p}}^{\lambda_1}$ and $\mathbf{x} = \tilde{\mathbf{x}}^{\lambda_1}$ are shown in (b) and $\Delta^1 x_{i,k} - F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ for $\mathbf{p} = \tilde{\mathbf{p}}^{\lambda_1}$ and $\mathbf{x} = \tilde{\mathbf{x}}^{\lambda_1}$ is shown in (d). In (a) and (b), $\Delta^1 x_{i,k}$ is shown with dashed lines and $F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ is shown with solid lines. Numerical solution errors appear to be uncorrelated in z for the noisy-data fit and highly correlated in z for the incomplete-model fit.

Paralleling inferences drawn from Figure 3.5, as is visible in Figure 3.6, for the fit of the traveling wave Bonny model to the noisy traveling-wave data, differences in $\Delta^1 x_{i,k}$ and $F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ for $\mathbf{p} = \tilde{\mathbf{p}}^{\lambda_1}$ and $\mathbf{x} = \tilde{\mathbf{x}}^{\lambda_1}$ appear to be mostly uncorrelated in z (with a Durbin-Watson statistic of 1.43), reflecting random error; whereas, for the fit of the incomplete traveling wave model to the synthetic traveling-wave data, differences in $\Delta^1 x_{i,k}$ and $F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ for $\mathbf{p} = \tilde{\mathbf{p}}^{\lambda_1}$ and $\mathbf{x} = \tilde{\mathbf{x}}^{\lambda_1}$ appear to be highly correlated in z (with a Durbin-Watson statistic of

0.50), reflecting modeling error. Furthermore, for the fit of the incomplete traveling wave model to the synthetic traveling-wave data, relatively large differences in $\Delta^1 x_{i,k}$ and $F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ for $\mathbf{p} = \tilde{\mathbf{p}}^{\lambda_1}$ and $\mathbf{x} = \tilde{\mathbf{x}}^{\lambda_1}$ occur near values of z where $\Delta^1 x_{i,k}$ changes rapidly, indicating that the modeling error could be Laplacian-dependent.

I provide an example that shows how a parameter from overlapping-niche descent differs over $\lambda \in (0, 1)$ for a fit of a complete model to noisy data and a fit of an incomplete model to noiseless data. For the example, I plot values of $\omega_{de,m}$ in $\tilde{\mathbf{p}}^{\lambda}$ for $\lambda \in \{\lambda_1, \lambda_2, \ldots, \lambda_{101}\}$ from the fit of the traveling wave Bonny model to the noisy traveling-wave data and the fit of the incomplete traveling wave model to the synthetic traveling-wave data in figure 3.7.

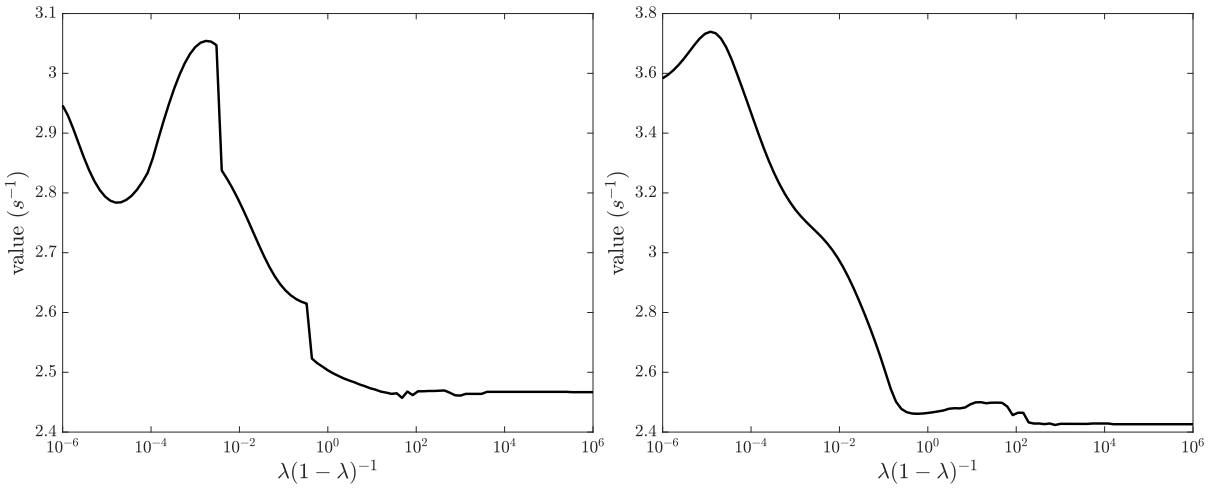


Figure 3.7: Parameter variation over $\lambda \in (0, 1)$ for noisy-data and incomplete-model fits. Values of $\omega_{de,m}$ in $\tilde{\mathbf{p}}^{\lambda}$ for $\lambda \in \{\lambda_1, \lambda_2, \ldots, \lambda_{101}\}$ from the fit of the traveling wave Bonny model to the noisy traveling-wave data are shown in (a). Values of $\omega_{de,m}$ in $\tilde{\mathbf{p}}^{\lambda}$ for $\lambda \in \{\lambda_1, \lambda_2, \ldots, \lambda_{101}\}$ from the fit of the incomplete traveling wave model to the synthetic traveling-wave data are shown in (b). I plot at $\lambda(1-\lambda)^{-1}$ on a log scale to distinguish values of $\omega_{de,m}$ in $\tilde{\mathbf{p}}^{\lambda}$ near $\lambda = 0$ and near $\lambda = 1$.

3.8 Overlapping-Niche Descent in Practice

In Sections 3.5, 3.6, and 3.7, I have simply shown overlapping-niche descent results. Here, I explicate overlapping-niche descent in practice. I continue my discussion for details pertaining to the implementation of overlapping-niche descent in practice in Section E.3. My discussion follows overlapping-niche descent in the fitting of the full Bonny model (3.1) to the synthetic traveling-wave-emergence data on a uniform grid with a grid refinement factor of 1, $n_{\Delta_t} n_{\Delta_s} {n_t}^{-1} {n_s}^{-1} = 1$ for n_{Δ_t} and n_{Δ_s} the number or temporal and spatial grid points and n_t and n_s the number of temporal and spatial data points.

3.8.1 Convergence

During overlapping-niche descent, I minimize $r(\mathbf{p}, \mathbf{x}; \lambda)$ over an array of λ values in $(0, 1)$ to find $\tilde{\mathbf{p}}^{\lambda}$ and $\tilde{\mathbf{x}}^{\lambda}$, which allows me to define the function $\tilde{r}(\lambda) = (1 - \lambda)\tilde{r}_{\mathrm{y}}(\lambda) + \lambda\tilde{r}_{\Delta x}(\lambda) = (1 - \lambda)r_{\mathrm{y}}(\tilde{\mathbf{p}}^{\lambda}, \tilde{\mathbf{x}}^{\lambda}) + \lambda r_{\Delta x}(\tilde{\mathbf{p}}^{\lambda}, \tilde{\mathbf{x}}^{\lambda})$, as defined in equation (2.23). I note that $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x}) = 0$ in $r(\mathbf{p}, \mathbf{x}; \lambda)$ for a grid refinement factor of 1, so the $\tilde{r}_{\hat{y}}(\lambda)$ term in $\tilde{r}(\lambda)$ is omitted here. To illustrate convergence in $\tilde{r}(\lambda)$ over generations of overlapping-niche descent, I plot the relative change in $\tilde{r}(\lambda)$ over each generation of overlapping-niche descent in Figure 3.8.

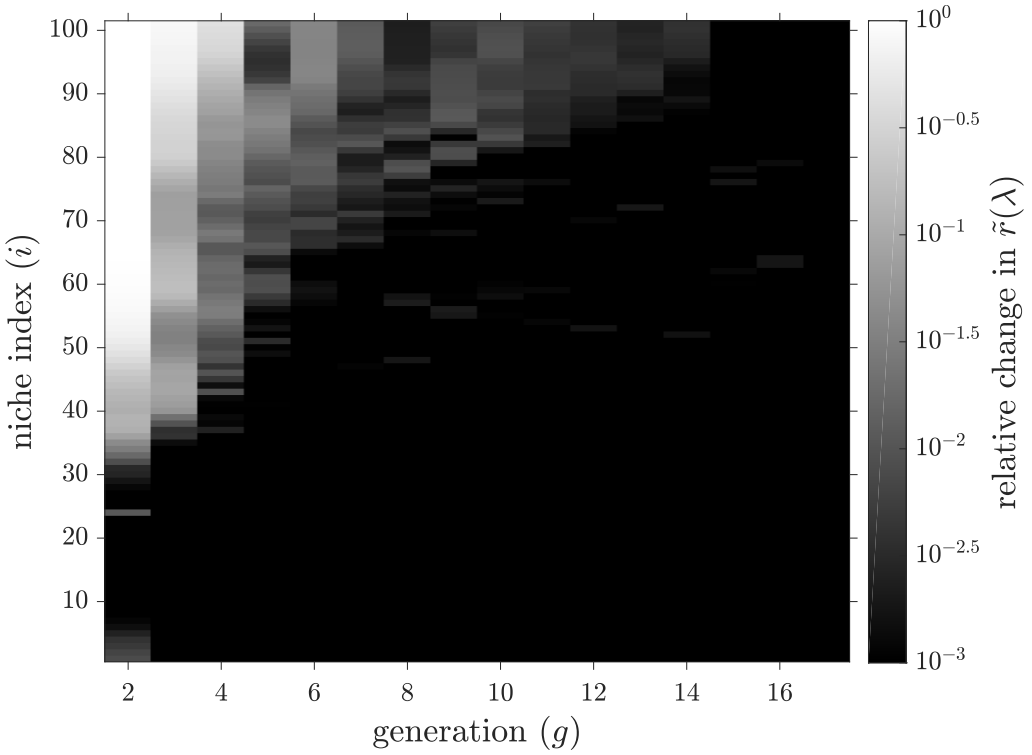


Figure 3.8: Convergence of $\tilde{r}(\lambda)$ during overlapping-niche descent. The relative change in $\tilde{r}(\lambda)$ over each generation of overlapping-niche descent is shown for niches defined by $\lambda \in \{\lambda_1, \lambda_2, \ldots, \lambda_{101}\}$. Values correspond to $\Delta r_{g,i,1}$, as defined in equation (C.1) of Section C.1, for niche index $i \in \{1, 2, \ldots, 101\}$ and generation $g > 2$. Generally, $\tilde{r}(\lambda)$ converges quickly for smaller λ and more slowly for larger λ.

As is visible in Figure 3.8, generally, $\tilde{r}(\lambda)$ converges sequentially in λ, in the order of increasing λ. Interestingly, as shown in Section E.3, although $\tilde{r}(\lambda)$ converges more readily for smaller λ, selection in overlapping-niche descent from a niche with a larger value of λ contributes to convergence in $\tilde{r}(\lambda)$ at least as much as selection from a niche with a smaller value of λ. Convergence in $\tilde{r}(\lambda)$ is coupled to convergence in $\tilde{r}_{\mathrm{y}}(\lambda)$ and $\tilde{r}_{\Delta x}(\lambda)$. To illustrate convergence in $\tilde{r}_{\mathrm{y}}(\lambda)$ and $\tilde{r}_{\Delta x}(\lambda)$, I plot the evolution of $\tilde{r}_{\mathrm{y}}(\lambda)$ and $\tilde{r}_{\Delta x}(\lambda)$ over generations of overlapping-niche descent in Figure 3.9.

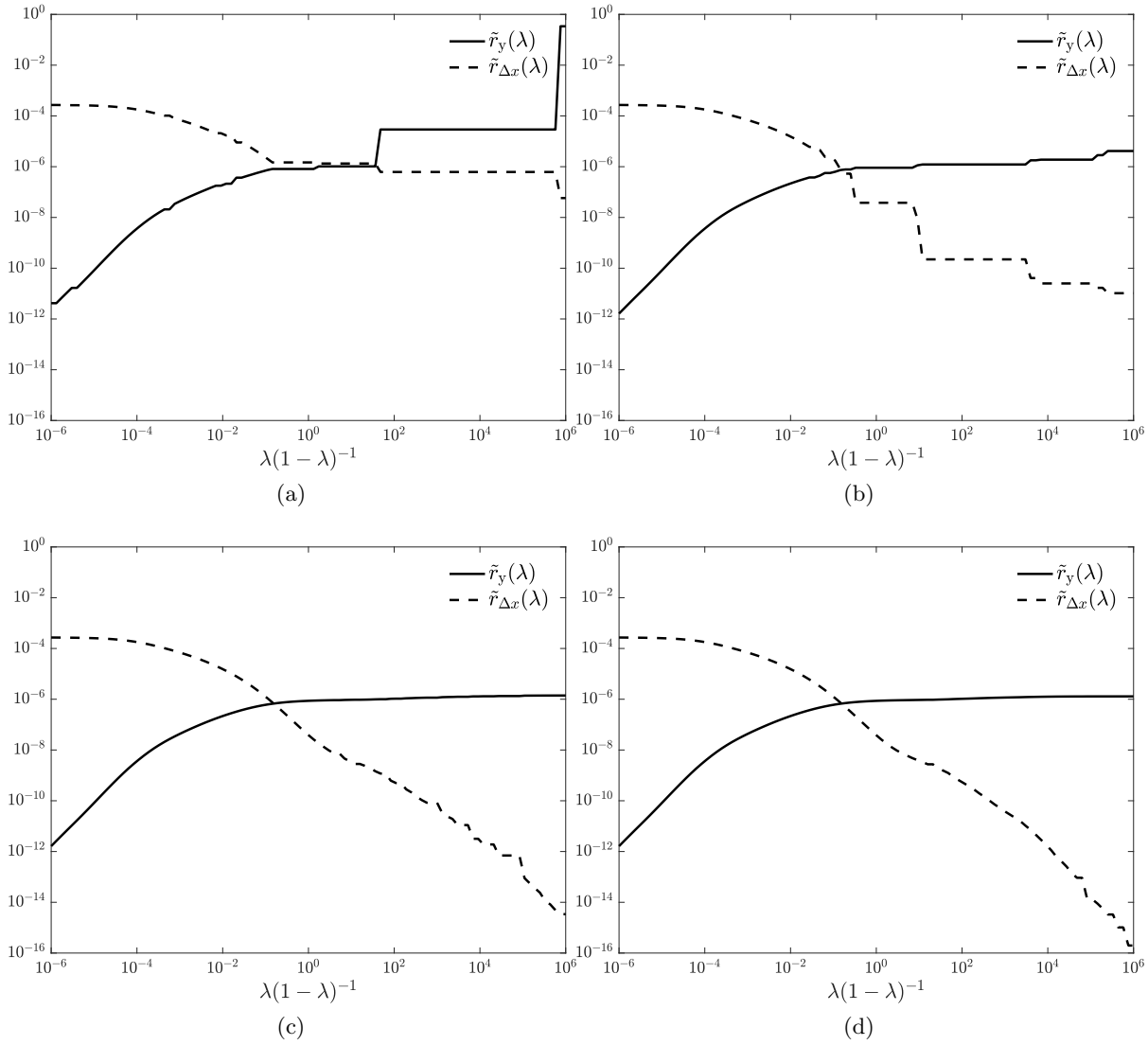


Figure 3.9: The evolution of $\tilde{r}_{\mathrm{y}}(\lambda)$ and $\tilde{r}_{\Delta x}(\lambda)$ over generations of overlapping-niche descent, for $\lambda \in \{\lambda_1, \lambda_2, \ldots, \lambda_{101}\}$. Values are shown in (a), (b), (c), and (d) for generations 1, 2, 5, and 17 of overlapping-niche descent. I plot at $\lambda(1-\lambda)^{-1}$ on a log scale to distinguish values of $\tilde{r}_{\mathrm{y}}(\lambda)$ and $\tilde{r}_{\Delta x}(\lambda)$ near $\lambda = 0$ and near $\lambda = 1$.

3.8.2 Consistency with the Conservation Principle and Integral Representations

For $\lambda \in (0, 1)$, the parameters and state values that minimize $r(\mathbf{p}, \mathbf{x}; \lambda)$, $\breve{\mathbf{p}}^{\lambda}$ and $\breve{\mathbf{x}}^{\lambda}$, allow me to define the function $\breve{r}(\lambda) = (1-\lambda)\breve{r}_{\mathrm{y}}(\lambda) + \lambda\breve{r}_{\Delta x}(\lambda) = (1-\lambda)r_{\mathrm{y}}(\breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda}) + \lambda r_{\Delta x}(\breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda})$, as defined in equation (2.18). I note that $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x}) = 0$ in $r(\mathbf{p}, \mathbf{x}; \lambda)$ for a grid refinement factor of 1, so the $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$ term in $\breve{r}(\lambda)$ is omitted here. $\breve{r}(\lambda)$, $\breve{r}_{\mathrm{y}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ satisfy a conservation

principle of the form:

$$2\int_0^1 \breve{r}(\lambda)d\lambda = \int_0^1 \breve{r}_{\mathrm{y}}(\lambda)d\lambda = \int_0^1 \breve{r}_{\Delta x}(\lambda)d\lambda, \tag{3.14}$$

as stipulated in equation (2.22). As such, I calculate and compare values of $2\int_0^1 \tilde{r}(\lambda)d\lambda$, $\int_0^1 \tilde{r}_{\mathrm{y}}(\lambda)d\lambda$, and $\int_0^1 \tilde{r}_{\Delta x}(\lambda)d\lambda$ to ensure that $\tilde{r}(\lambda)$, $\tilde{r}_{\mathrm{y}}(\lambda)$, and $\tilde{r}_{\Delta x}(\lambda)$ are consistent with the conservation principle. In doing so, I numerically calculate integral values by integrating piecewise cubic spline interpolants of integrand values with not-a-knot end conditions. I plot values of $2\int_0^1 \tilde{r}(\lambda)d\lambda$, $\int_0^1 \tilde{r}_{\mathrm{y}}(\lambda)d\lambda$, and $\int_0^1 \tilde{r}_{\Delta x}(\lambda)d\lambda$ for each generation of overlapping-niche descent in Figure 3.10.

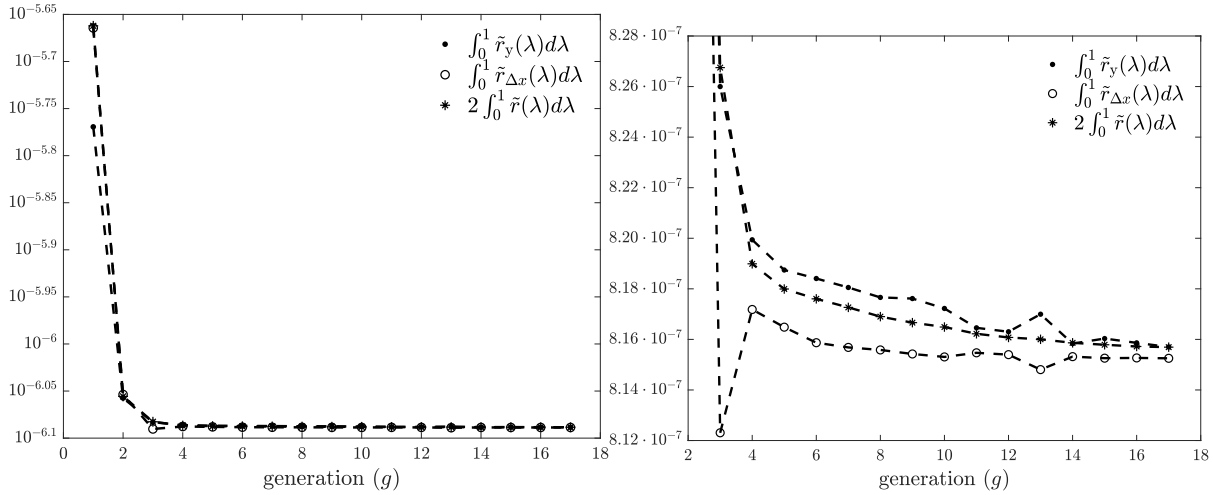


Figure 3.10: Consistency in conservation of $\tilde{r}(\lambda)$, $\tilde{r}_{\mathrm{y}}(\lambda)$, and $\tilde{r}_{\Delta x}(\lambda)$. Values of $2\int_0^1 \tilde{r}(\lambda)d\lambda$, $\int_0^1 \tilde{r}_{\mathrm{y}}(\lambda)d\lambda$, and $\int_0^1 \tilde{r}_{\Delta x}(\lambda)d\lambda$ are shown for generations 1-17 in (a) and for generations 3-17 (for a more focused view) in (b). Dashed lines are shown to delineate values. $2\int_0^1 \tilde{r}(\lambda)d\lambda$, $\int_0^1 \tilde{r}_{\mathrm{y}}(\lambda)d\lambda$, and $\int_0^1 \tilde{r}_{\Delta x}(\lambda)d\lambda$ converge to similar values over generations.

As is visible in Figure 3.10, $2\int_0^1 \tilde{r}(\lambda)d\lambda$, $\int_0^1 \tilde{r}_{\mathrm{y}}(\lambda)d\lambda$, and $\int_0^1 \tilde{r}_{\Delta x}(\lambda)d\lambda$ converge to similar values over generations of overlapping-niche descent, indicating that, ultimately, $\tilde{r}(\lambda)$, $\tilde{r}_{\mathrm{y}}(\lambda)$, and $\tilde{r}_{\Delta x}(\lambda)$ are fairly consistent with the conservation principle.

$\breve{r}_{\mathrm{y}}(\lambda)$ and $\breve{r}_{\Delta x}(\lambda)$ admit integral representations of limit values:

$$\lim_{\lambda\to 0^+} \breve{r}_{\Delta x}(\lambda) = \int_0^1 \lambda^{-2}\breve{r}_{\mathrm{y}}(\lambda)d\lambda, \tag{3.15}$$

$$\lim_{\lambda\to 1^-} \breve{r}_{\mathrm{y}}(\lambda) = \int_0^1 (1-\lambda)^{-2}\breve{r}_{\Delta x}(\lambda)d\lambda, \tag{3.16}$$

as stipulated in equation (2.25). As such, I calculate and compare values of $\tilde{r}_{\Delta x}(\lambda_1)$ with $\int_0^1 \lambda^{-2}\tilde{r}_{\mathrm{y}}(\lambda)d\lambda$ and $\tilde{r}_{\mathrm{y}}(\lambda_{101})$ with $\int_0^1 (1-\lambda)^{-2}\tilde{r}_{\Delta x}(\lambda)d\lambda$ to ensure that $\tilde{r}_{\mathrm{y}}(\lambda)$ and $\tilde{r}_{\Delta x}(\lambda)$ are

consistent with the integral representations of limit values. In doing so, I numerically calculate integral values by integrating piecewise cubic spline interpolants of integrand values with not-a-knot end conditions. I find that $\tilde{r}_{\Delta x}(\lambda_1) \approx 2.70 \cdot 10^{-4}$ and $\int_0^1 \lambda^{-2}\tilde{r}_{\mathrm{y}}(\lambda)d\lambda \approx 2.72 \cdot 10^{-4}$ for all generations of overlapping-niche descent. I plot values of $\tilde{r}_{\mathrm{y}}(\lambda_{101})$ and $\int_0^1 (1-\lambda)^{-2}\tilde{r}_{\Delta x}(\lambda)d\lambda$ for each generation of overlapping-niche descent in Figure 3.11.

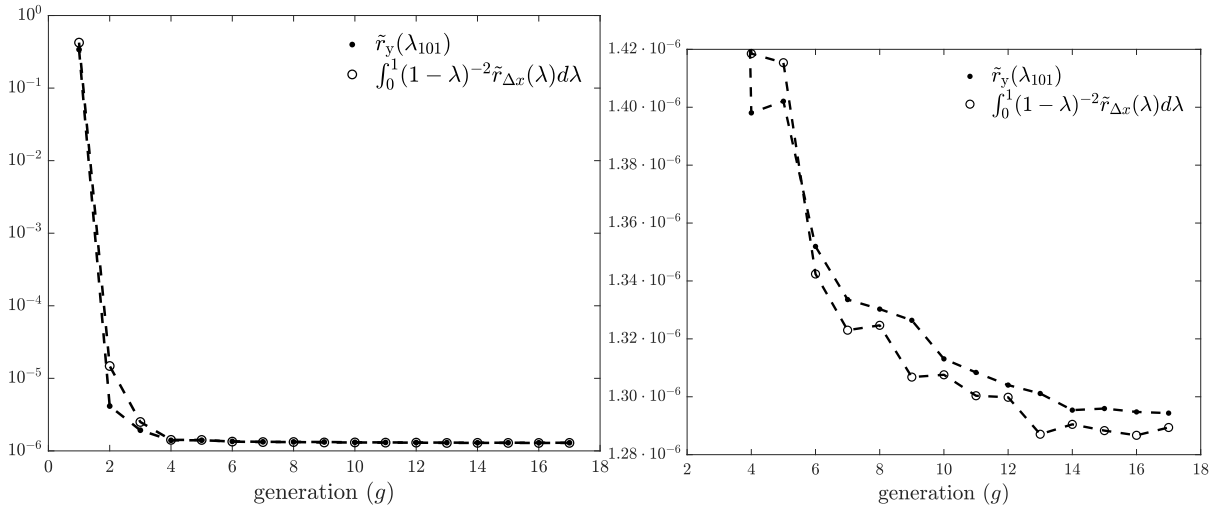


Figure 3.11: Consistency in the integral representations of $\lim_{\lambda \to 1^-} \tilde{r}_{\mathrm{y}}(\lambda)$. Values of $\tilde{r}_{\mathrm{y}}(\lambda_{101})$ and $\int_0^1 (1-\lambda)^{-2}\tilde{r}_{\Delta x}(\lambda)d\lambda$ are shown for generations 1-17 in (a) and for generations 4-17 (for a more focused view) in (b). Dashed lines are shown to delineate values. $\tilde{r}_{\mathrm{y}}(\lambda_{101})$ and $\int_0^1 (1-\lambda)^{-2}\tilde{r}_{\Delta x}(\lambda)d\lambda$ converge to similar values over generations.

As is visible in Figure 3.11, $\tilde{r}_{\mathrm{y}}(\lambda_{101})$ and $\int_0^1 (1-\lambda)^{-2}\tilde{r}_{\Delta x}(\lambda)d\lambda$ converge to similar values over generations of overlapping-niche descent. I conclude that $\tilde{r}_{\mathrm{y}}(\lambda)$ and $\tilde{r}_{\Delta x}(\lambda)$ are fairly consistent with the integral representations of limit values.

I explore how $\tilde{r}_{\mathrm{y}}(\lambda)$ and $\tilde{r}_{\Delta x}(\lambda)$ contribute to the integral representations of limit values for different values of λ. Given the weighting λ^{-2}, it would appear that $\int_0^1 \lambda^{-2}\tilde{r}_{\mathrm{y}}(\lambda)d\lambda$ depends on $\tilde{r}_{\mathrm{y}}(\lambda)$ most heavily for λ near 0. However, $\tilde{r}_{\mathrm{y}}(\lambda)$ is smallest for λ near 0. Similarly, given the weighting $(1-\lambda)^{-2}$, it would appear that $\int_0^1 (1-\lambda)^{-2}\tilde{r}_{\Delta x}(\lambda)d\lambda$ depends on $\tilde{r}_{\Delta x}(\lambda)$ most heavily for λ near 1. However, $\tilde{r}_{\Delta x}(\lambda)$ is smallest for λ near 1. Thus, the extent to which the integral representations of limit values depend on $\tilde{r}_{\mathrm{y}}(\lambda)$ and $\tilde{r}_{\Delta x}(\lambda)$ for different values of λ is unclear. For clarification, I plot cumulative integrals of $\int_0^1 \lambda^{-2}\tilde{r}_{\mathrm{y}}(\lambda)d\lambda$ and $\int_0^1 (1-\lambda)^{-2}\tilde{r}_{\Delta x}(\lambda)d\lambda$ in Figure 3.12.

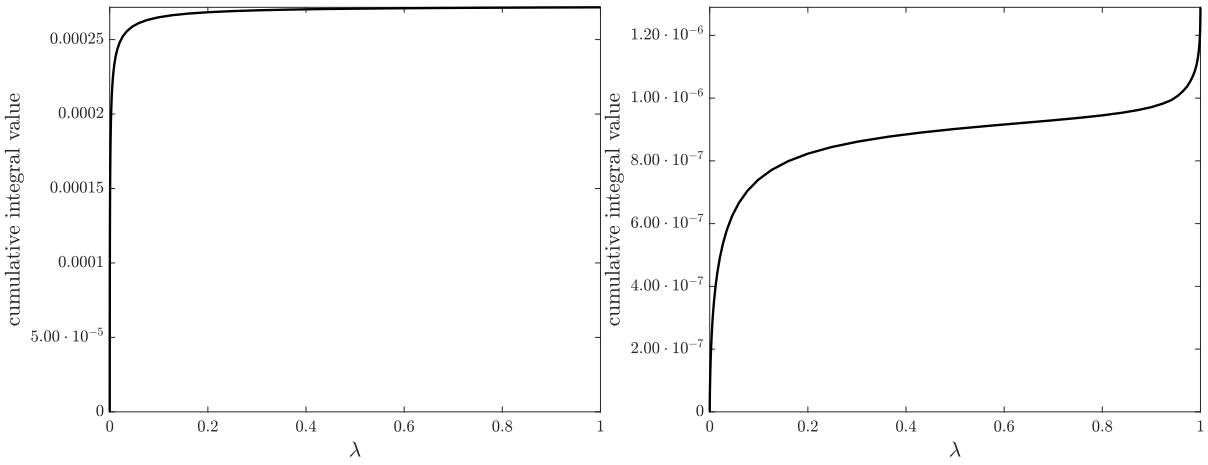


Figure 3.12: Cumulative integral representations of limit values. The cumulative integral of $\int_0^1 \lambda^{-2}\tilde{r}_{\mathrm{y}}(\lambda)d\lambda$ is shown in (a). The cumulative integral of $\int_0^1 (1-\lambda)^{-2}\tilde{r}_{\Delta x}(\lambda)d\lambda$ is shown in (b). $\int_0^1 \lambda^{-2}\tilde{r}_{\mathrm{y}}(\lambda)d\lambda$ and $\int_0^1 (1-\lambda)^{-2}\tilde{r}_{\Delta x}(\lambda)d\lambda$ are rooted most heavily in small to intermediate λ.

As is visible in Figure 3.12, $\int_0^1 \lambda^{-2}\tilde{r}_{\mathrm{y}}(\lambda)d\lambda$ depends on $\tilde{r}_{\mathrm{y}}(\lambda)$ most heavily for small λ. Also, $\int_0^1 (1-\lambda)^{-2}\tilde{r}_{\Delta x}(\lambda)d\lambda$ depends on $\tilde{r}_{\Delta x}(\lambda)$ most heavily for small to intermediate λ and for λ near 1. Interestingly, despite the weighting of $(1-\lambda)^{-2}$, $\int_0^1 (1-\lambda)^{-2}\tilde{r}_{\Delta x}(\lambda)d\lambda$ depends on $\tilde{r}_{\Delta x}(\lambda)$ more heavily for small to intermediate λ than for λ near 1. Thus, $\int_0^1 \lambda^{-2}\tilde{r}_{\mathrm{y}}(\lambda)d\lambda$ and $\int_0^1 (1-\lambda)^{-2}\tilde{r}_{\Delta x}(\lambda)d\lambda$ are rooted in $\tilde{r}_{\mathrm{y}}(\lambda)$ and $\tilde{r}_{\Delta x}(\lambda)$ for small to intermediate λ, which, as $\tilde{r}(\lambda)$ converges most readily for small to intermediate λ, implies that $\tilde{r}_{\mathrm{y}}(\lambda)$ and $\tilde{r}_{\Delta x}(\lambda)$ for small to intermediate λ provide a robust basis for the integral representations of limit values.

3.9 Discussion

In this chapter, I tested my method on synthetic data and a system of first order ordinary differential equations, a system of second order ordinary differential equations, and a system of partial differential equations. I found that my method accurately identified the optimal data-fitting numerical solution and its parameters in all three contexts. I compared the performance of my method to that of an analogous numerical-integration-based method, and found that my method identified the optimal data-fitting numerical solution more robustly than the analogous numerical-integration-based method, while requiring significantly less time to do so. I also explored an example where my method informed modeling insufficiencies and potential model improvements for an incomplete variant of a model. Finally, I showed that my optimization routine converged to values that were consistent with my derived conservation principles and integral representations.

Chapter 4

Fitting Models of the Min System to Time-Course Data

4.1 Introduction

The *Escherichia coli* Min system is one of the simplest known biological systems that demonstrates diverse complex dynamic behavior or transduces local interactions into a global signal. As such, the Min system is currently one of the most reduced model systems for understanding such behaviors. Various mathematical models of the Min system show behaviors that are qualitatively similar to dynamic behaviors of the Min system that have been observed in experiments, but no model has been quantitatively compared to time-course data. In this chapter, I briefly summarize extracting time-course data for model fitting from experimental measurements of the Min system and fit established and novel biochemistry-based models to the time-course data using my method, which I developed in Chapter 2 and tested in Chapter 3. Comparing models to time-course data allows me to make precise distinctions between biochemical assumptions in the various models. My modeling and fitting supports a novel model that accounts for MinE's previously unmodeled dual role as a stabilizer and an inhibitor of MinD membrane binding. It suggests that a regular, ordered, stability-switching mechanism underlies the emergent, dynamic behavior of the Min system.

4.2 Choosing and Processing Data to Simplify Fitting

For data fitting, I use *in vitro* data from the experiments of Ivanov and Mizuuchi and focus on regions that are as close to spatially homogeneous as possible. *In vitro* data poses fewer challenges for time course comparison than *in vivo* data: *in vitro* geometry is a simple two-dimensional plane, whereas *in vivo* geometry is a relatively complex three-dimensional rod shape; *in vitro* measurements map a two-dimensional process to a two-dimensional image, whereas *in vivo* measurements map a three-dimensional process to a two-dimensional image; *in vitro* data is less coarse than *in vivo* data, as the spatial scale of pattern formation is larger in *in vitro* experiments than in *in vivo* experiments; and *in vitro* behavior is less susceptible to stochastic effects than *in vivo* behavior because *in vitro* experiments employ significantly higher concentrations of proteins than *in vivo* experiments.

In the Ivanov and Mizuuchi *in vitro* experiments, buffer was rapidly flowed atop a supported

lipid bilayer to induce spatially uniform concentrations of reaction components in the buffer. On the supported lipid bilayer, densities of MinD and MinE, which were measured using total internal reflection microscopy (TIRF), oscillated near-homogeneously in space before forming into traveling waves [38]. Details of the Ivanov and Mizuuchi experiments are described throughout Appendix F. Deterministic models of the Min system are generally systems of partial differential equations that describe how protein concentrations change in space and time. In Section F.4.1, I show that the global behavior of a near-homogeneous process is described to leading order by a system of ordinary differential equations, the spatially-homogeneous reduction of the system's partial differential equation description. For the Min system, the spatially-homogeneous reduction is a description of how local reactions change Min protein concentrations in time. As such, fitting an ordinary differential equation model of the Min system to near-homogeneous time-course data provides a direct comparison of the model's reaction-based outcomes and experimental observations. Kindly, Ivanov and Mizuuchi have shared their data with me.

4.2.1 Extracting Spatially Near-Homogeneous Data

The Ivanov and Mizuuchi data requires some preprocessing before being able to extract near-homogeneous data from it. Fluorescence intensities of fluorescently labeled MinD and MinE are not spatially aligned in the Ivanov and Mizuuchi data. I align the Ivanov and Mizuuchi data using the cross-correlation of similarly shaped structures in MinD and MinE fluorescence intensity profiles. Details of data alignment are described in Section F.2. After aligning the Ivanov and Mizuuchi data, I flatten the aligned data to correct for variability in MinD and MinE fluorescent intensities from Gaussian illumination in microscopy. Details of data flattening are described in Section F.3.2. The conversion from MinE fluorescence intensity to MinE density was not directly measured in the Ivanov and Mizuuchi experiments. From bulk concentrations of MinD and MinE and from properties of evanescent waves in total internal reflection microscopy (TIRF), I calculate conversions from flattened MinD and MinE fluorescent intensities to MinD and MinE densities. Details of calculating conversions from fluorescence intensities to densities are described in Section F.3.3.

To extract near-homogeneous data, first, I find the MinD and MinE density data that is the least inhomogeneous within a disk of 1000 pixels for all times within a spatially near-homogeneous oscillation in the Ivanov and Mizuuchi data. Then, I calculate mean values of MinD and MinE densities within the disk at each time. Details of extracting near-homogeneous data are described in Section F.4.2. Near-homogeneous MinD and MinE density data is shown in Figure 4.1.

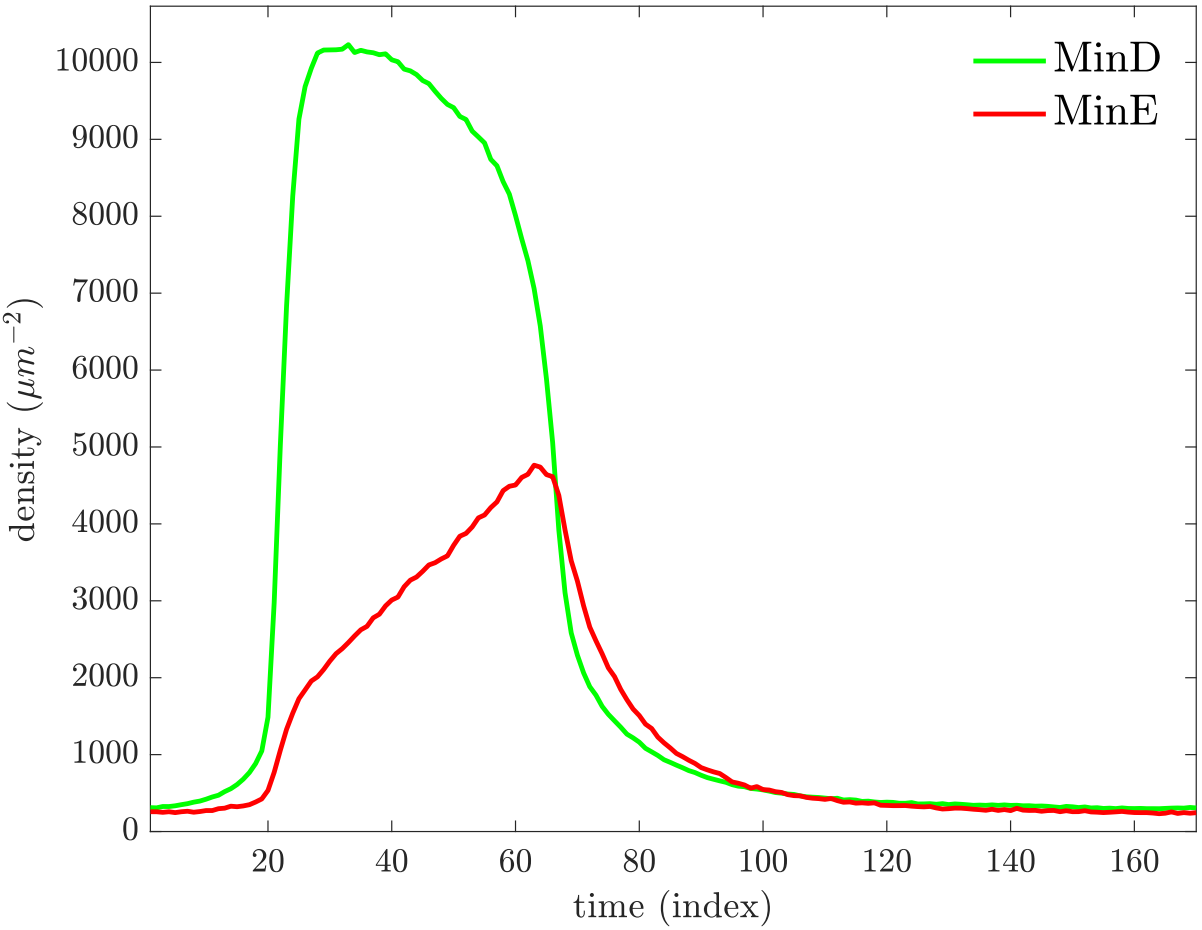


Figure 4.1: Near-homogeneous MinD and MinE density data. Data is extracted from measurements made by Ivanov and Mizuuchi [38], in which densities of MinD and MinE oscillate near-homogeneously in space on a supported lipid bilayer.

If I had included error bars in figure 4.1 representing standard errors of the mean, they would not be visually distinguishable from the data. Details of errors in near-homogeneous MinD and MinE density data are described in Section F.4.3. Interestingly, as detailed in Section F.4.3, I find that errors in near-homogeneous MinD and MinE density data are related to values of near-homogeneous MinD and MinE density data by power laws.

4.3 Fitting Models to the Near-Homogeneous Data

Local reactions somehow coordinate membrane binding and unbinding of the Min proteins in a way that collectively generates the emergent, dynamic, global behavior of the Min system. Fitting a model, in the form of an ordinary differential equation, to the near-homogeneous time-course data provides me with a direct measure of how well the model's reaction-based outcomes describe the near-homogeneous data. As such, fitting models with a variety of proposed Min-system reaction mechanisms to the near-homogeneous data allows me to make precise

distinctions between biochemical assumptions in the various models, helping to unravel the specifics of the local Min-system reaction mechanism.

I fit models to the near-homogeneous data using my homotopy-minimization method, which I developed in Chapter 2 and tested in Chapter 3, to find optimal data-fitting numerical solutions for models. Some quantities related to model parameter values have been measured in experiments. As such, during fitting, I restrict values of parameters using the experimentally measured values, to confine some parameters to biologically realistic values. Details of fitting are described in Section 4.4, and details of parameter restrictions based on experimental measurements are described in Section 4.4.3.

4.3.1 Modeling and Fitting Brief

Among previously published models, the Bonny model [4] has demonstrated the most diverse array of dynamic behaviors that are qualitatively similar to experimental observations of the Min system *in vivo* and *in vitro*. To begin my investigation, in Section 4.3.2, I modify the Bonny model to account for the details of the experimental protocol used by Ivanov and Mizuuchi and fit the Modified Bonny Model to the near-homogeneous data. Then, I extend the Bonny model to include new reactions based on experiments, data, and postulate and fit the Extended Bonny Model to the near-homogeneous data. MinE has generally been thought to act as an inhibitor of MinD membrane binding. The Modified Bonny Model and the Extended Bonny Model treat MinE as such. Recently, however, it has been shown that MinE can act to both stabilize and inhibit MinD membrane binding [73]. I build on the Extended Bonny Model, in Section 4.3.3, to develop two models that could account for MinE's dual role as a stabilizer and an inhibitor of MinD membrane binding, the Symmetric Activation Model and the Asymmetric Activation Model, and fit them to the near-homogeneous data. Ultimately, I find that my Asymmetric Activation Model fits the near-homogeneous data best, suggesting, as described in Section 4.3.4, that a regular, ordered, stability-switching mechanism underlies the emergent, dynamic behavior of the Min system.

4.3.2 Models in Which MinE Acts Only as an Inhibitor

Biochemical analysis has characterized how MinD and MinE interact with the membrane and each other. In the cytosol, MinD monomers bind to ATP and form dimers ([34], [81]) that sandwich two ATP molecules [76]. MinD dimers bind to the phospholipid membrane ([28], [34], [32], [42], [81], [45]) and cooperatively recruit other MinD dimers to the membrane [42]. MinE dimers bind to MinD dimers on the membrane ([28], [44]), undergoing a conformational change that allows MinE dimers to bind to the membrane [52]. MinE dimers stimulate ATPase activity in bound MinD dimers, causing MinD dimers to separate and dissociate from the membrane ([31], [28], [34], [42]), leaving MinE dimers temporarily bound to the membrane ([27], [66], [73]). Most mathematical models of the Min system, including the Bonny model, are based on a subset of the aforementioned set of reactions.

The Modified Bonny Model

The Bonny model [4] demonstrates an array of dynamic behavior that is qualitatively similar to many experimental observations of the Min system, including stochastic pole-to-pole switching in short cells, regular pole-to-pole oscillations in mid-sized cells, oscillation splitting in growing cells, regular pole-to-midcell oscillations in long cells, end-to-end oscillations in thick cells, and spiral waves on a supported lipid bilayer. The Bonny model consists of three membrane-bound states, c_d, c_{de}, and c_e, corresponding to the membrane-bound concentrations of MinD dimers, MinE dimers bound to MinD dimers, and MinE dimers respectively. Normally, the Bonny model includes two cytosolic states, c_D and c_E, corresponding to the concentrations of MinD and MinE dimers in the cytosol respectively. Because of spatially uniform concentrations of reaction components in the buffer of the Ivanov and Mizuuchi experiments, I modify the Bonny model such that c_D and c_E are constant. The Bonny model is a system of partial differential equations. In the case of spatial homogeneity, the Bonny model reduces to a system of ordinary differential equations. As such, for correspondence with the near-homogeneous data, I reduced the Bonny model to a system of ordinary differential equations under spatially homogeneous conditions:

$$\frac{dc_d}{dt} = (\omega_{D\to d} + \omega^d_{D\to d}c_d)(c_{\max} - c_d - c_{de})/c_{\max} - \omega_{E,d\to de}c_d - \omega_{d,e\to de}c_d c_e, \tag{4.1a}$$

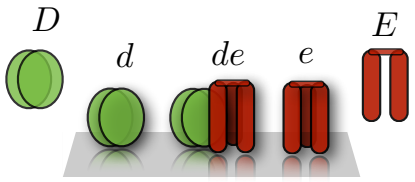
$$\frac{dc_{de}}{dt} = \omega_{E,d\to de}c_d + \omega_{d,e\to de}c_d c_e - \omega_{de\to D,E}c_{de} - \omega_{de\to D,e}c_{de}, \tag{4.1b}$$

$$\frac{dc_e}{dt} = -\omega_{d,e\to de}c_d c_e + \omega_{de\to D,e}c_{de} - \omega_{e\to E}c_e, \tag{4.1c}$$

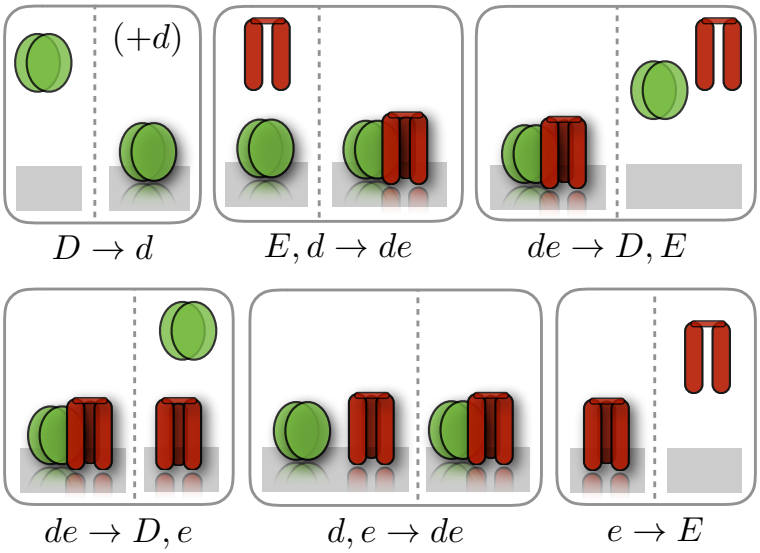
where $c_{\max}$ is the saturation concentration of MinD dimers on the membrane and $\omega_{u,v\to x,y}$ denotes the reaction rate of c_u and c_v converting into c_x and c_y, for $u, v, x, y \in \{\emptyset, D, E, d, de, e\}$. When $\omega_{u,v\to x,y}$ has a superscript it indicates facilitation of the reaction by the superscripted species. I note that $\omega_{D\to d}$ and $\omega^d_{D\to d}$ have a multiplicative factor of c_D built into them and $\omega_{E,d\to de}$ has a multiplicative factor of c_E built into it. Also, I note that I have changed parameter notation in the Modified Bonny Model for consistency with upcoming models. In terms of the original parameter notation, as used in Chapter 3, $\omega_{D\to d} = \omega_D \cdot c_D$, $\omega^d_{D\to d} = \omega_{dD} \cdot c_D$, $\omega_{E,d\to de} = \omega_E \cdot c_E$, $\omega_{de\to D,E} = \omega_{de,c}$, $\omega_{de\to D,e} = \omega_{de,m}$, $\omega_{e\to E} = \omega_e$, and $\omega_{d,e\to de} = \omega_{ed}$. The Modified Bonny Model (4.1) is depicted in Figure 4.2.

$\omega_{D\to d}$	$\omega^{d}_{D\to d}$	$\omega_{E,d\to de}$	$\omega_{de\to D,E}$	$\omega_{de\to D,e}$	$\omega_{d,e\to de}$	$\omega_{e\to E}$

(a)



(b)



(c)

Figure 4.2: The Modified Bonny Model. Parameters characterizing reactions are shown in (a). State variables are matched to protein states in (b). Reactions are depicted in (c). In (c), reactants are shown on the left and products are shown on the right of panels. (+) indicates facilitation in a reaction by the indicated species.

I fit the Modified Bonny Model (4.1) to the near-homogeneous data, as described in Section 4.4. The resulting fit, state values, and parameter values are shown in Figure 4.3, Figure 4.4, and Table 4.1 respectively.

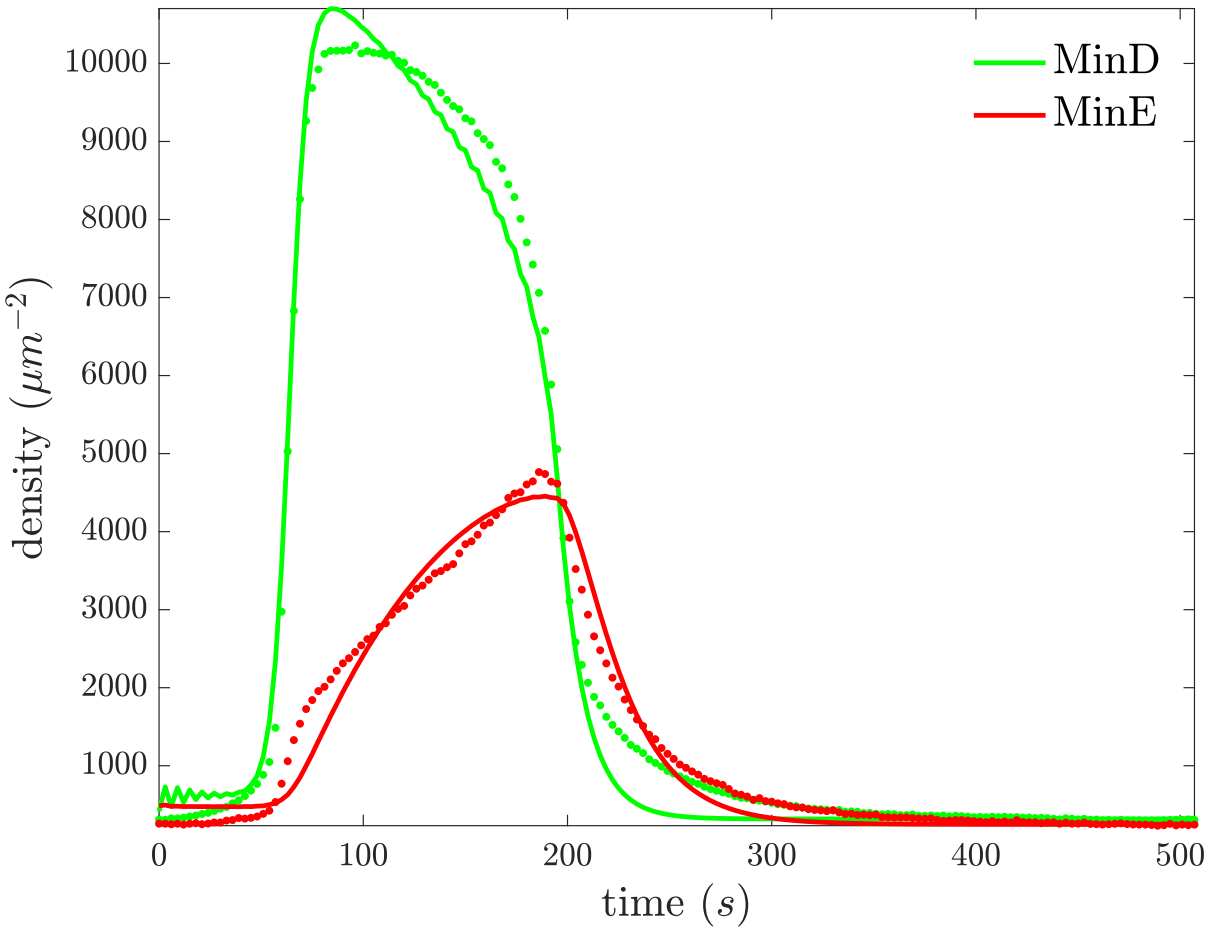


Figure 4.3: The fit of the Modified Bonny Model to the near-homogeneous data. Data is shown with points and model values are shown with lines.

As is visible in Figure 4.3, the Modified Bonny Model admits pulses in MinD and MinE that are qualitatively similar in width and height to the MinD and MinE pulses of the near-homogeneous data. However, the Modified Bonny Model is visibly an incomplete description of the dynamical system underlying the near-homogeneous data. I seek experimentally-based alterations of the Modified Bonny Model that allow for a better description of the dynamical system underlying the near-homogeneous data.

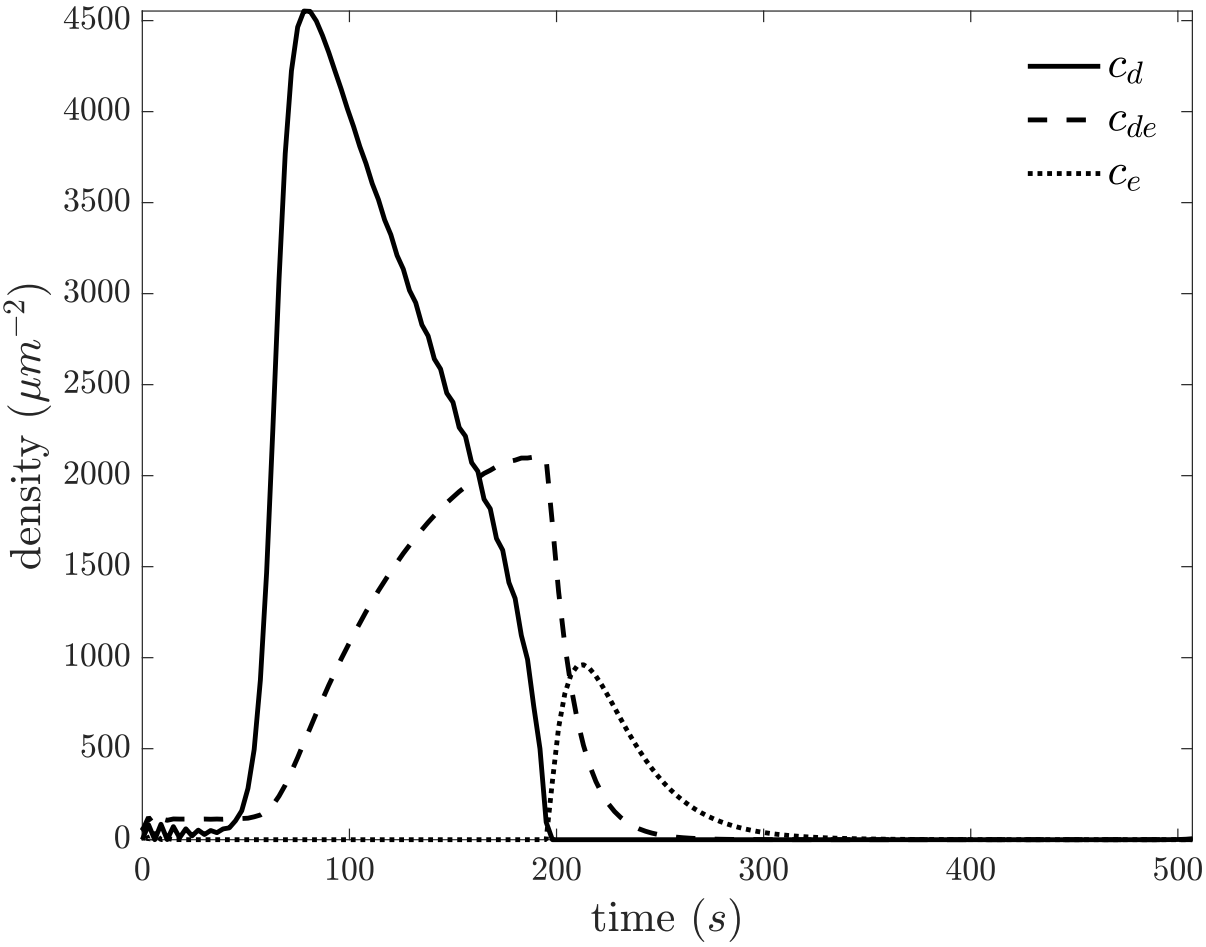


Figure 4.4: States from the fit of the Modified Bonny Model to the near-homogeneous data.

parameter	value	95% confidence interval	units
C_d	$3.18 \cdot 10^2$	$[3.18 \cdot 10^2, 3.18 \cdot 10^2]$	μm^{-2}
C_e	$2.49 \cdot 10^2$	$[2.26 \cdot 10^2, 2.49 \cdot 10^2]$	μm^{-2}
$c_{\max}$	$5.48 \cdot 10^3$	$[5.47 \cdot 10^3, 5.65 \cdot 10^3]$	μm^{-2}
$\omega_{D \to d}$	$6.31 \cdot 10^{-6}$	$[8.48 \cdot 10^{-3}, 4.67 \cdot 10^{-1}]$	$\mu m^{-2}\ s^{-1}$
$\omega^d_{D \to d}$	$2.47 \cdot 10^{-1}$	$[2.20 \cdot 10^{-1}, 2.61 \cdot 10^{-1}]$	s^{-1}
$\omega_{E,d \to de}$	$6.23 \cdot 10^{-3}$	$[5.73 \cdot 10^{-3}, 6.59 \cdot 10^{-3}]$	s^{-1}
$\omega_{d,e \to de}$	$1.23 \cdot 10^0$	$[1.21 \cdot 10^0, 1.25 \cdot 10^0]$	$\mu m^2\ s^{-1}$
$\omega_{de \to D,E}$	$2.43 \cdot 10^{-3}$	$[1.55 \cdot 10^{-3}, 3.10 \cdot 10^{-3}]$	s^{-1}
$\omega_{de \to D,e}$	$7.81 \cdot 10^{-2}$	$[7.20 \cdot 10^{-2}, 8.16 \cdot 10^{-2}]$	s^{-1}
$\omega_{e \to E}$	$4.54 \cdot 10^{-2}$	$[4.03 \cdot 10^{-2}, 4.96 \cdot 10^{-2}]$	s^{-1}

Table 4.1: Parameters from the fit of the Modified Bonny Model to the near-homogeneous data. C_d and C_e are fitted data-motivated shifts in observable state values, data preprocessing parameters described in Section F.4.4 that correspond to the constant concentrations of monomers in the bulk and persistently bound monomers on the membrane, for MinD and MinE respectively. Details of calculating confidence intervals are described in Section 4.4.5.

The Extended Bonny Model

The Bonny model assumes that c_d, but not by c_{de}, recruits c_D to the membrane. In MinD and MinE bursts on a supported lipid bilayer *in vitro*, increasing the MinE concentration increases both the membrane-binding rate of MinD and the peak membrane density of MinD [73]. Thus, the binding of MinE to MinD on the supported lipid bilayer does not seem to suppress MinD's ability to recruit bulk MinD to the supported lipid bilayer. As such, I extend the Modified Bonny Model to allow c_{de} to recruit c_D to the membrane.

The Bonny model assumes that neither c_{de} or c_e recruits c_E to bind to c_d on the membrane. In fact, I know of no model that incorporates the facilitated recruitment of cytosolic MinE to bind to free MinD on the membrane. Without facilitated recruitment, a constant concentration of cytosolic MinE binds to free MinD on the membrane with rate proportional to the concentration of free MinD on the membrane. As shown in Figure 4.1, MinD decreases from a density of $\sim 10,000\ \mu m^{-2}$ to a density of $\sim 7,000\ \mu m^{-2}$ while MinE increases at a roughly constant rate from a density of $\sim 2,000\ \mu m^{-2}$ to a density of $\sim 5,000\ \mu m^{-2}$. Thus, the MinE recruitment rate does not trail off with decreasing MinD (as seen in the Bonny model in Figure 4.3), which suggests that cytosolic MinE may be recruited to bind to free MinD on the membrane with facilitation. Thus, I extend the Modified Bonny Model, allowing c_{de} and c_e to recruit c_E to bind to c_d.

Despite MinE's long-established role in inducing hydrolysis by MinD, *in vitro* experiments show that MinD rapidly dissociates from the supported lipid bilayer in the absence of MinE [38]. Furthermore, the residence time of MinD on the supported lipid bilayer increases from 11 s to at least 40.71 s as the concentration of MinD increases from 0.275 μM to 1.1 μM [44]. Without information on the mechanism of stabilization, I phenomenologically model the rate

"constant" of spontaneous c_d dissociation by a reverse hill function,

$$\frac{\omega_{d\to D} c_s^{n_s}}{c_s^{n_s} + (c_d + c_{de})^{n_s}}, \tag{4.2}$$

where $\omega_{d\to D}$ is the basal spontaneous dissociation rate of c_d, c_s is the half-max stabilization concentration of $c_d + c_{de}$, and n_s is the Hill coefficient. Thus, c_d decreases from spontaneous c_d dissociation with rate

$$\frac{\omega_{d\to D} c_s^{n_s} c_d}{c_s^{n_s} + (c_d + c_{de})^{n_s}}. \tag{4.3}$$

In the absence of MinE *in vitro*, buffer flowed atop a MinD-saturated supported lipid bilayer reveals that, initially, MinD spontaneously dissociates from the supported lipid bilayer at a roughly constant rate [73]. Thus, from rate (4.3), for relatively large c_d,

$$\frac{\omega_{d\to D} c_s^{n_s} c_d}{c_s^{n_s} + c_d^{n_s}} \approx k, \tag{4.4}$$

for some constant k, which occurs only if

$$c_s << c_d \text{ and } n_s = 1. \tag{4.5}$$

Therefore, I extend the Modified Bonny Model, allowing the spontaneous dissociation of c_d, at the rate given by (4.2) with $n_s = 1$.

I also consider the possibility of reversible reactions in the Extended Bonny Model. In traveling waves of Min proteins on a supported lipid bilayer *in vitro*, MinE residence times are at least 1.3 times as long as MinD residence times in all portions of the traveling waves [44]. Thus, I do not consider reactions where MinE spontaneously dissociates from MinD on the supported lipid bilayer. I do consider the other reverse reaction, the reaction of c_{de} splitting into c_d and c_e.

I extend the Modified Bonny Model (4.1) such that

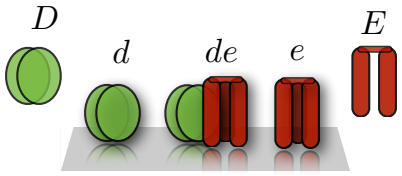
$$\begin{aligned}
\frac{dc_d}{dt} &= (\omega_{D\to d} + \omega^d_{D\to d} c_d + \omega^{de}_{D\to d} c_{de})(c_{\max} - c_{\bar{d}} - c_d - c_{de})/c_{\max} \\
&\quad - (\omega_{E,d\to de} + \omega^{de}_{E,d\to de} c_{de} + \omega^{e}_{E,d\to de} c_e) c_d - \omega_{d,e\to de} c_d c_e + \omega_{de\to d,e} c_{de} \\
&\quad - \omega_{d\to D} c_s c_d/(c_s + c_{\bar{d}} + c_d + c_{de}), && \text{(4.6a)} \\
\frac{dc_{de}}{dt} &= (\omega_{E,d\to de} + \omega^{de}_{E,d\to de} c_{de} + \omega^{e}_{E,d\to de} c_e) c_d + \omega_{d,e\to de} c_d c_e \\
&\quad - \omega_{de\to D,E} c_{de} - \omega_{de\to D,e} c_{de} - \omega_{de\to d,e} c_{de}, && \text{(4.6b)} \\
\frac{dc_e}{dt} &= -\omega_{d,e\to de} c_d c_e + \omega_{de\to d,e} c_{de} + \omega_{de\to D,e} c_{de} - \omega_{e\to E} c_e, && \text{(4.6c)}
\end{aligned}$$

where $c_{\max}$ is the saturation concentration of MinD dimers on the membrane, $c_{\bar{d}}$ is the constant concentration of persistently bound MinD dimers on the membrane, an experimental artifact

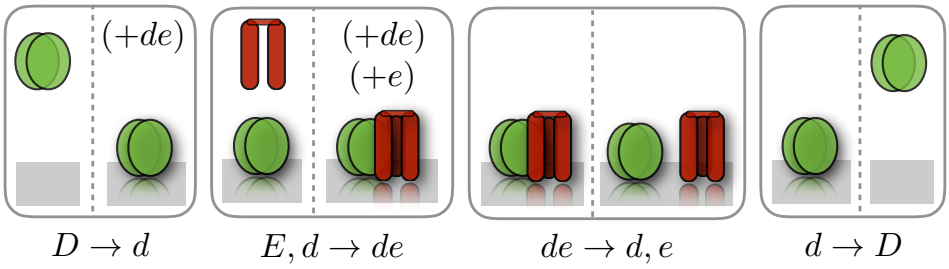
discussed in Section F.4.4, and $\omega_{u,v\to x,y}$ denotes the reaction rate of c_u and c_v converting into c_x and c_y, for $u, v, x, y \in \{\emptyset, D, E, d, de, e\}$. When $\omega_{u,v\to x,y}$ has a superscript it indicates facilitation of the reaction by the superscripted species. I note that $\omega^z_{D\to d}$ has a multiplicative factor of c_D built into it for $z \in \{\emptyset, d, de\}$ and $\omega^z_{E,d\to de}$ has a multiplicative factor of c_E built into it for $z \in \{\emptyset, de, e\}$. I also note that I do not explicitly include $c_{\bar{d}}$ in the Modified Bonny Model (4.1), as it is absorbed by $c_{\max}$, $\omega_{D\to d}$, and $\omega^d_{D\to d}$, but must include it in the Extended Bonny Model because the structure of equation (4.6a) doesn't allow it to be rescaled away. The Extended Bonny Model (4.6) is depicted in Figure 4.5.

$\omega_{D\to d}$	$\omega^d_{D\to d}$	$\bullet\omega^{de}_{D\to d}$	$\omega_{E,d\to de}$	$\bullet\omega^{de}_{E,d\to de}$	$\bullet\omega^{e}_{E,d\to de}$
$\omega_{d,e\to de}$	$\omega_{de\to D,E}$	$\omega_{de\to D,e}$	$\bullet\omega_{de\to d,e}$	$\bullet\omega_{d\to D}$	$\omega_{e\to E}$

(a)



(b)



(c)

Figure 4.5: The Extended Bonny Model. Parameters characterizing reactions are shown in (a). A parameter that characterizes a reaction that is not included in the Modified Bonny Model is shown with a bullet (•) and the corresponding reaction is depicted in (c). All reactions from the Modified Bonny Model are included in the Extended Bonny Model. State variables are matched to protein states in (b). In (c), reactants are shown on the left and products are shown on the right of panels. (+) indicates facilitation in a reaction by the indicated species.

I fit the Extended Bonny Model (4.6) to the near-homogeneous data, as described in Section 4.4. The resulting fit, state values, and parameter values are shown in Figure 4.6, Figure 4.7, and Table 4.2 respectively.

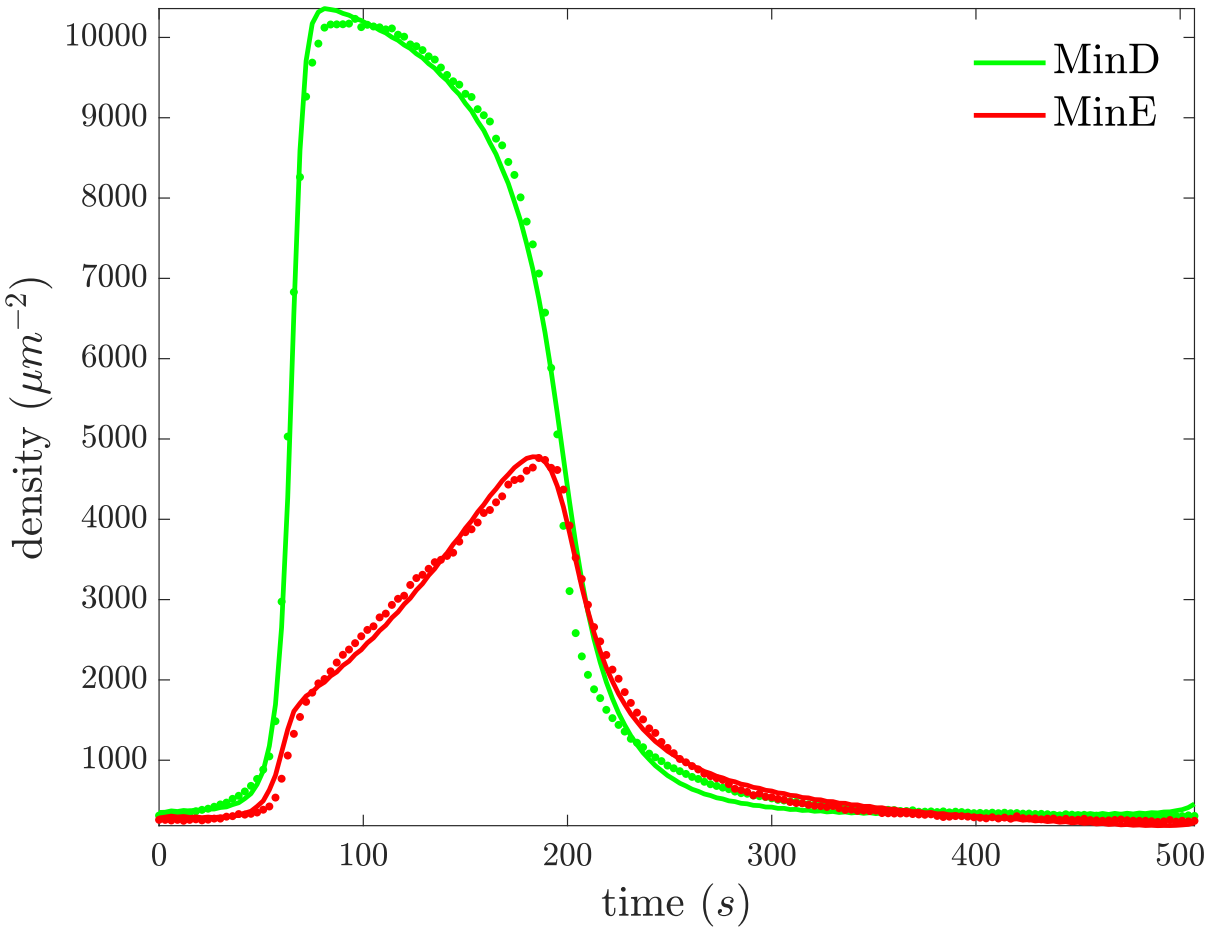


Figure 4.6: The fit of the Extended Bonny Model to the near-homogeneous data. Data is shown with points and model values are shown with lines. The Extended Bonny Model fits the near-homogeneous data appreciably better than the Modified Bonny Model (compare to Figure 4.3).

Comparing Figures 4.3 and 4.6, the Extended Bonny Model describes the near-homogeneous data visibly better than the Modified Bonny Model. Quantitatively, χ^2, the weighted sum of squared residuals, from the Modified Bonny Model is 3.53 times larger than χ^2 from the Extended Bonny Model, and the value of AIC, the Akaike information criterion, from the Modified Bonny Model is 419 units larger than the value of AIC from the Extended Bonny Model. The Akaike information criterion is a measure of a model's ability to fit data that accounts for the number of parameters in the model, based on information theory. For a set of competing models, the model with the minimum AIC value is considered the best model.

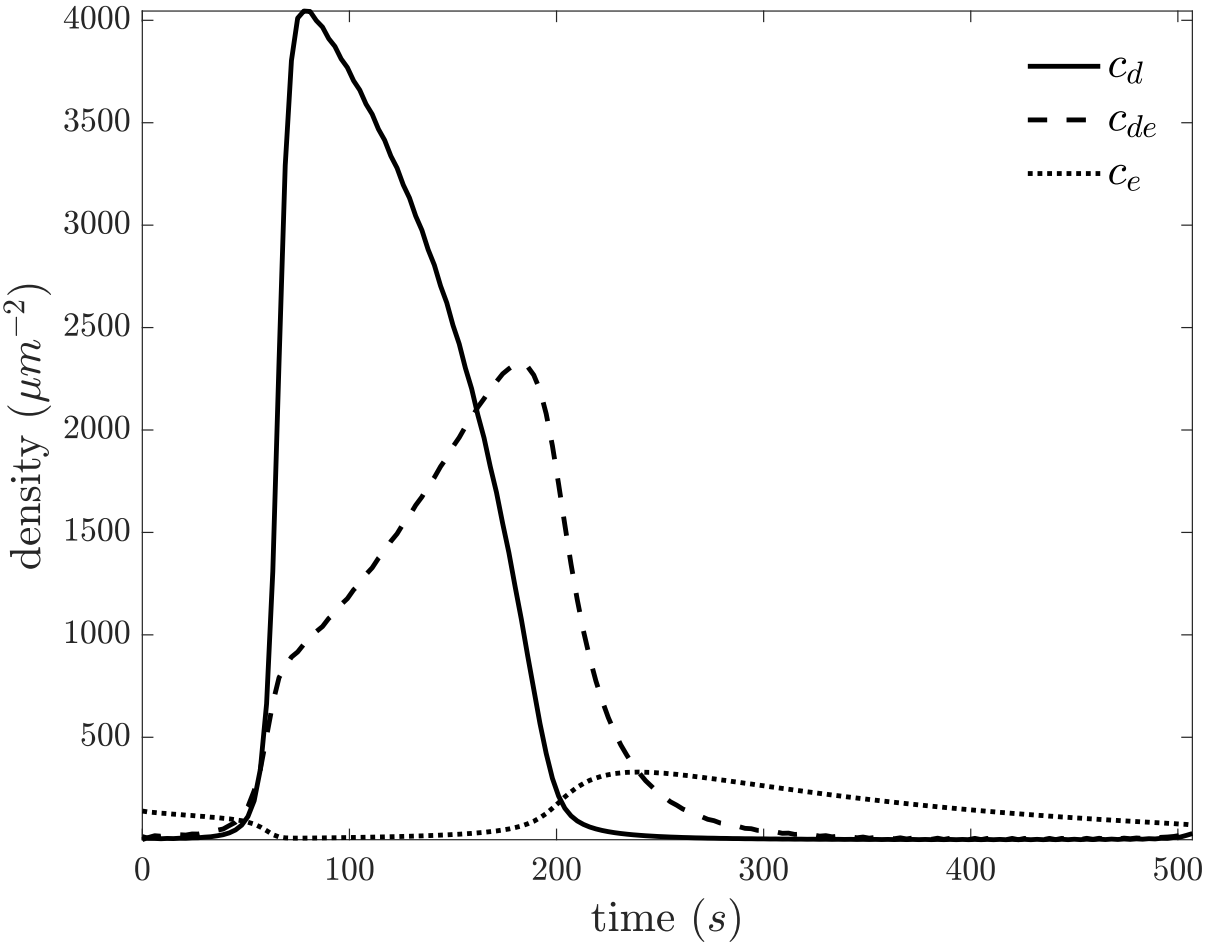


Figure 4.7: States from the fit of the Extended Bonny Model to the near-homogeneous data.

parameter	value	95% confidence interval	units
C_d	$3.18 \cdot 10^2$	$[2.74 \cdot 10^2, 3.18 \cdot 10^2]$	μm^{-2}
C_e	$0.00 \cdot 10^0$	$[0.00 \cdot 10^0, 2.52 \cdot 10^1]$	μm^{-2}
$c_{\bar{d}}$	$7.50 \cdot 10^{-8}$	$[0.00 \cdot 10^0, 1.18 \cdot 10^1]$	μm^{-2}
$c_{\max}$	$5.38 \cdot 10^3$	$[5.32 \cdot 10^3, 5.48 \cdot 10^3]$	μm^{-2}
c_s	$2.95 \cdot 10^1$	$[1.03 \cdot 10^1, 1.89 \cdot 10^2]$	μm^{-2}
$\omega_{D \to d}$	$3.26 \cdot 10^{-2}$	$[0.00 \cdot 10^0, 1.26 \cdot 10^{-1}]$	$\mu m^{-2}\ s^{-1}$
$\omega^{d}_{D \to d}$	$4.88 \cdot 10^{-1}$	$[4.45 \cdot 10^{-1}, 5.35 \cdot 10^{-1}]$	s^{-1}
$\omega^{de}_{D \to d}$	$6.48 \cdot 10^{-2}$	$[5.65 \cdot 10^{-2}, 7.69 \cdot 10^{-2}]$	s^{-1}
$\omega_{E,d \to de}$	$9.08 \cdot 10^{-5}$	$[0.00 \cdot 10^0, 5.96 \cdot 10^{-4}]$	s^{-1}
$\omega^{de}_{E,d \to de}$	$0.00 \cdot 10^0$	$[0.00 \cdot 10^0, 6.15 \cdot 10^{-7}]$	$\mu m^2\ s^{-1}$
$\omega^{e}_{E,d \to de}$	$3.68 \cdot 10^{-3}$	$[3.32 \cdot 10^{-3}, 4.37 \cdot 10^{-3}]$	$\mu m^2\ s^{-1}$
$\omega_{d,e \to de}$	$5.17 \cdot 10^{-4}$	$[4.59 \cdot 10^{-4}, 6.02 \cdot 10^{-4}]$	$\mu m^2\ s^{-1}$
$\omega_{d \to D}$	$3.15 \cdot 10^{-1}$	$[2.17 \cdot 10^{-1}, 3.15 \cdot 10^{-1}]$	s^{-1}
$\omega_{de \to D,E}$	$1.13 \cdot 10^{-1}$	$[1.04 \cdot 10^{-1}, 1.24 \cdot 10^{-1}]$	s^{-1}
$\omega_{de \to D,e}$	$1.75 \cdot 10^{-2}$	$[1.54 \cdot 10^{-2}, 1.89 \cdot 10^{-2}]$	s^{-1}
$\omega_{de \to d,e}$	$1.10 \cdot 10^{-6}$	$[0.00 \cdot 10^0, 7.72 \cdot 10^{-4}]$	s^{-1}
$\omega_{e \to E}$	$6.22 \cdot 10^{-3}$	$[5.61 \cdot 10^{-3}, 6.92 \cdot 10^{-3}]$	s^{-1}

Table 4.2: Parameters from the fit of the Extended Bonny Model to the near-homogeneous data. C_d and C_e are as described in Table 4.1. Details of calculating confidence intervals are described in Section 4.4.5.

To determine how individual reactions affect the Extended Bonny Model's ability to describe the near-homogeneous data, individually, I remove non-necessary reactions from the Extended Bonny Model and fit the resulting model to the near-homogeneous data, as described in Section 4.4. Results are shown in Table 4.3.

Null parameter	$\chi^2/\chi^2_\emptyset$	$\text{AIC} - \text{AIC}_\emptyset$
$\omega^d_{D\to d}$	$2.85 \cdot 10^1$	$1.14 \cdot 10^3$
$\omega^{de}_{D\to d}$	$1.32 \cdot 10^0$	$8.53 \cdot 10^1$
$\omega^{de}_{E,d\to de}$	$1.00 \cdot 10^0$	$-2.05 \cdot 10^0$
$\omega^{e}_{E,d\to de}$	$2.62 \cdot 10^0$	$3.26 \cdot 10^2$
$\omega_{d,e\to de}$	$7.96 \cdot 10^0$	$6.99 \cdot 10^2$
$\omega_{d\to D}$	$1.10 \cdot 10^0$	$2.56 \cdot 10^1$
$\omega_{de\to D,E}$	$2.63 \cdot 10^0$	$3.27 \cdot 10^2$
$\omega_{de\to D,e}$	$1.03 \cdot 10^0$	$5.89 \cdot 10^0$
$\omega_{de\to d,e}$	$1.00 \cdot 10^0$	$-4.06 \cdot 10^0$

Table 4.3: Removed-reaction fits of the Extended Bonny Model to the near-homogeneous data. A removed reaction is characterized by a null parameter. χ^2 is the the weighted sum of squared residuals, and AIC is the Akaike information criterion. $\chi^2_\emptyset$ and $\text{AIC}_\emptyset$ are the values of χ^2 and AIC from the Extended Bonny Model without a removed reaction. $\chi^2/\chi^2_\emptyset$ and $\text{AIC} - \text{AIC}_\emptyset$ measure the affect of removing the reaction characterized by the null parameter from the Extended Bonny Model on its ability to fit the near-homogeneous data; larger values of $\chi^2/\chi^2_\emptyset$ and $\text{AIC} - \text{AIC}_\emptyset$ correspond to a larger decrease in fitting ability.

As shown in Table 4.3, the reactions characterized by the parameters $\omega^d_{D\to d}$, $\omega^{de}_{D\to d}$, $\omega^e_{E,d\to de}$, $\omega_{d,e\to de}$, and $\omega_{de\to D,E}$ each significantly ($\text{AIC} - \text{AIC}_\emptyset > 50$) affect the Extended Bonny Model's ability to describe the near-homogeneous data. Of these, $\omega^{de}_{D\to d}$ and $\omega^e_{E,d\to de}$ are not included in the Modified Bonny Model. Notably, much of the Extended Bonny Model's ability to describe the near-homogeneous data better than the Modified Bonny Model comes from the recruitment of c_E to bind to c_d by c_e (through $\omega^e_{E,d\to de}$). The fit of the $\omega^e_{E,d\to de}$-null Extended Bonny Model to the near-homogeneous data is shown in Figure 4.8.

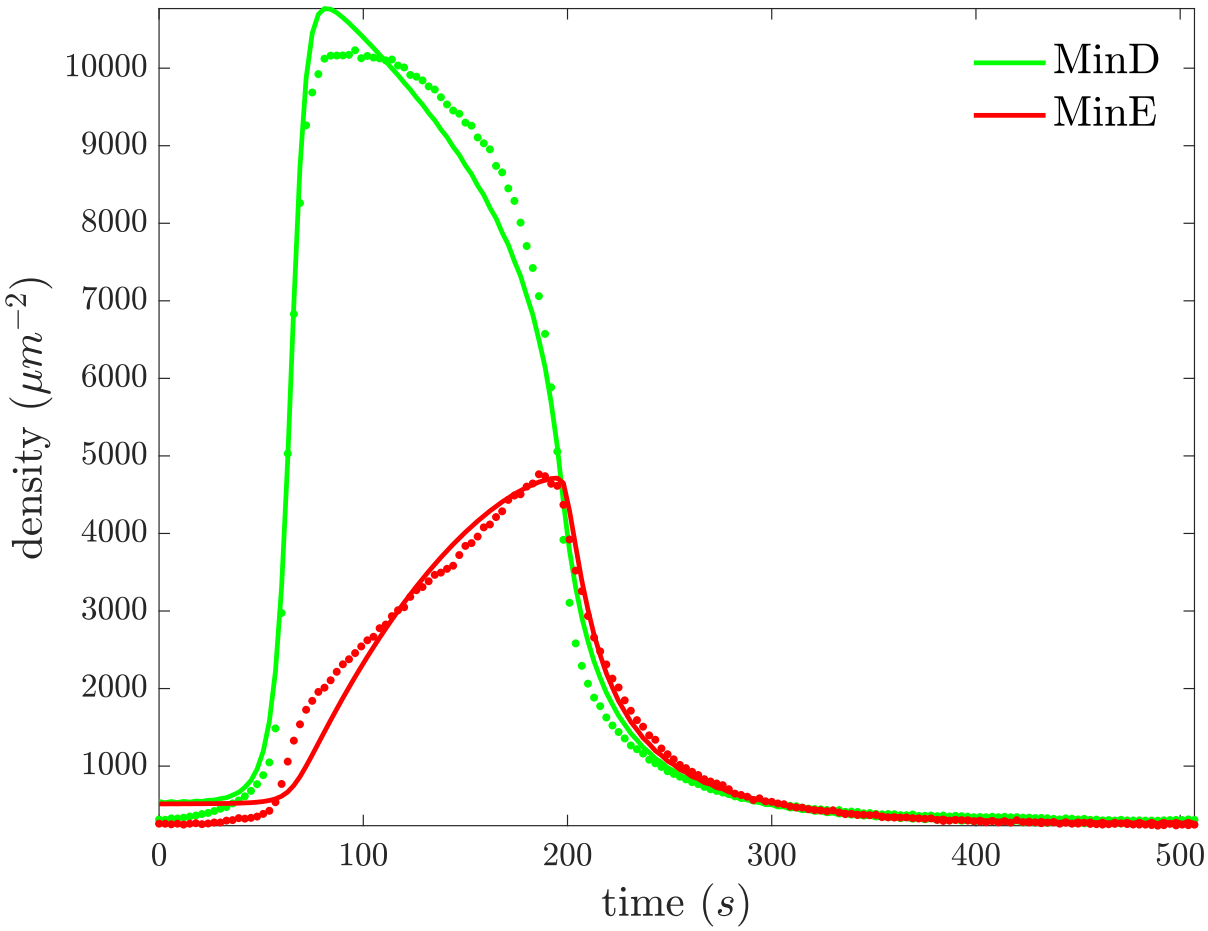


Figure 4.8: The fit of the $\omega^e_{E,d\to de}$-null Extended Bonny Model to the near-homogeneous data. Data is shown with points and model values are shown with lines. Removing the reaction characterized by the parameter $\omega^e_{E,d\to de}$ from the Extended Bonny Model significantly reduces its ability to fit the near-homogeneous data (compare to Figure 4.6).

The Extended Bonny Model fits the near-homogeneous data better than the Modified Bonny Model, but it is still an incomplete description of the dynamical system underlying the near-homogeneous data, as can be seen in the visible deviations between data and model which are larger than the noise in the data. Also, the Extended Bonny Model does not account for experimental observations that MinE can act as both a stabilizer and an inhibitor of MinD membrane binding.

4.3.3 Models in Which MinE Acts as Both a Stabilizer and an Inhibitor

With buffer and MinE flowed atop a MinD-saturated supported lipid bilayer, initially, MinD dissociates from the supported lipid bilayer more slowly than with buffer alone. Later, as concentrations of MinD and MinE on the supported lipid bilayer approach each other, MinD dissociates from the supported lipid bilayer more quickly than with buffer alone [73]. Thus,

MinE seems to act as both a stabilizer and an inhibitor of MinD membrane binding. Accordingly, the experimentally measured rate of MinD ATPase activity is sigmoidal in the concentration of MinE, showing switch-like behavior [73]. I build on the Extended Bonny Model to develop two models, the Symmetric Activation Model and the Asymmetric Activation Model, based on experiments and postulate that could account for MinE's dual role as a stabilizer and an inhibitor of MinD membrane binding, and fit them to the near-homogeneous data.

The Symmetric Activation Model

It has been proposed that MinD and MinE may form a stable complex, with one MinE dimer bound to one subunit of the MinD dimer, and an unstable complex, with one MinE dimer bound to each subunit of the MinD dimer ([73], [54]). This symmetric activation could explain MinE's dual role as a stabilizer and an inhibitor of MinD membrane binding. I build on the Extended Bonny Model (4.6) to develop the Symmetric Activation Model.

My Symmetric Activation Model consists of four supported-lipid-bilayer-bound states, c_d, c_{de}, c_{ede}, and c_e, respectively corresponding to the supported-lipid-bilayer-bound concentrations of MinD dimers, MinD dimers bound to one MinE dimer, MinD dimers bound to two MinE dimers, and MinE dimers. For symmetric activation, c_{de} is the stable state and c_{ede} is the unstable state. Thus, I exclude reactions from the Symmetric Activation Model where c_{de} dissociates from the supported lipid bilayer, and I include reactions in the Symmetric Activation Model where c_{ede} dissociates from the supported lipid bilayer. In the Symmetric Activation Model, without reason for restriction, I allow c_d, c_{de}, and c_{ede} to recruit c_D, the concentration of bulk MinD dimers, to the supported lipid bilayer, and I allow c_{de}, c_{ede}, and c_e to recruit c_E, the concentration of bulk MinE dimers, to bind to c_d and c_{de}. As in the Extended Bonny Model, I only consider reactions where MinE does not spontaneously dissociate from MinD on the supported lipid bilayer. Thus, in the Symmetric Activation Model, I include forward and reverse bimolecular reactions of c_d, c_{de}, c_{ede}, and c_e with all products in $\{c_d, c_{de}, c_{ede}, c_e\}$. Also, as in the Extended Bonny Model, in the Symmetric Activation Model, I include the spontaneous dissociation of c_d with stabilization

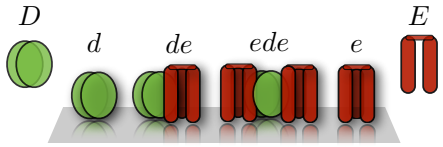
by supported-lipid-bilayer-bound MinD dimers. I define the Symmetric Activation Model:

$$\begin{aligned}
\frac{dc_d}{dt} &= (\omega_{D\to d} + \omega^{d}_{D\to d}c_d + \omega^{de}_{D\to d}c_{de} + \omega^{ede}_{D\to d}c_{ede})(c_{\max} - c_{\bar{d}} - c_d - c_{de} - c_{ede})/c_{\max} \\
&\quad - (\omega_{E,d\to de} + \omega^{de}_{E,d\to de}c_{de} + \omega^{ede}_{E,d\to de}c_{ede} + \omega^{e}_{E,d\to de}c_e)c_d \\
&\quad - \omega_{d,e\to de}c_d c_e + \omega_{de\to d,e}c_{de} - \omega_{d,ede\to de,de}c_d c_{ede} + \omega_{de,de\to d,ede}c_{de}^2 \\
&\quad - \omega_{d\to D}c_s c_d/(c_s + c_{\bar{d}} + c_d + c_{de} + c_{ede}), \qquad (4.7a) \\
\frac{dc_{de}}{dt} &= (\omega_{E,d\to de} + \omega^{de}_{E,d\to de}c_{de} + \omega^{ede}_{E,d\to de}c_{ede} + \omega^{e}_{E,d\to de}c_e)c_d \\
&\quad - (\omega_{E,de\to ede} + \omega^{de}_{E,de\to ede}c_{de} + \omega^{ede}_{E,de\to ede}c_{ede} + \omega^{e}_{E,de\to ede}c_e)c_{de} \\
&\quad + \omega_{d,e\to de}c_d c_e - \omega_{de\to d,e}c_{de} + 2\omega_{d,ede\to de,de}c_d c_{ede} - 2\omega_{de,de\to d,ede}c_{de}^2 \\
&\quad - \omega_{de,e\to ede}c_{de}c_e + \omega_{ede\to de,e}c_{ede}, \qquad (4.7b) \\
\frac{dc_{ede}}{dt} &= (\omega_{E,de\to ede} + \omega^{de}_{E,de\to ede}c_{de} + \omega^{ede}_{E,de\to ede}c_{ede} + \omega^{e}_{E,de\to ede}c_e)c_{de} \\
&\quad - \omega_{d,ede\to de,de}c_d c_{ede} + \omega_{de,de\to d,ede}c_{de}^2 + \omega_{de,e\to ede}c_{de}c_e - \omega_{ede\to de,e}c_{ede} \\
&\quad - \omega_{ede\to D,e,e}c_{ede} - \omega_{ede\to E,D,e}c_{ede} - \omega_{ede\to E,D,E}c_{ede}, \qquad (4.7c) \\
\frac{dc_e}{dt} &= \omega_{de\to d,e}c_{de} - \omega_{d,e\to de}c_d c_e - \omega_{de,e\to ede}c_{de}c_e \\
&\quad + \omega_{ede\to de,e}c_{ede} + \omega_{ede\to E,D,e}c_{ede} + 2\omega_{ede\to D,e,e}c_{ede} - \omega_{e\to E}c_e, \qquad (4.7d)
\end{aligned}$$

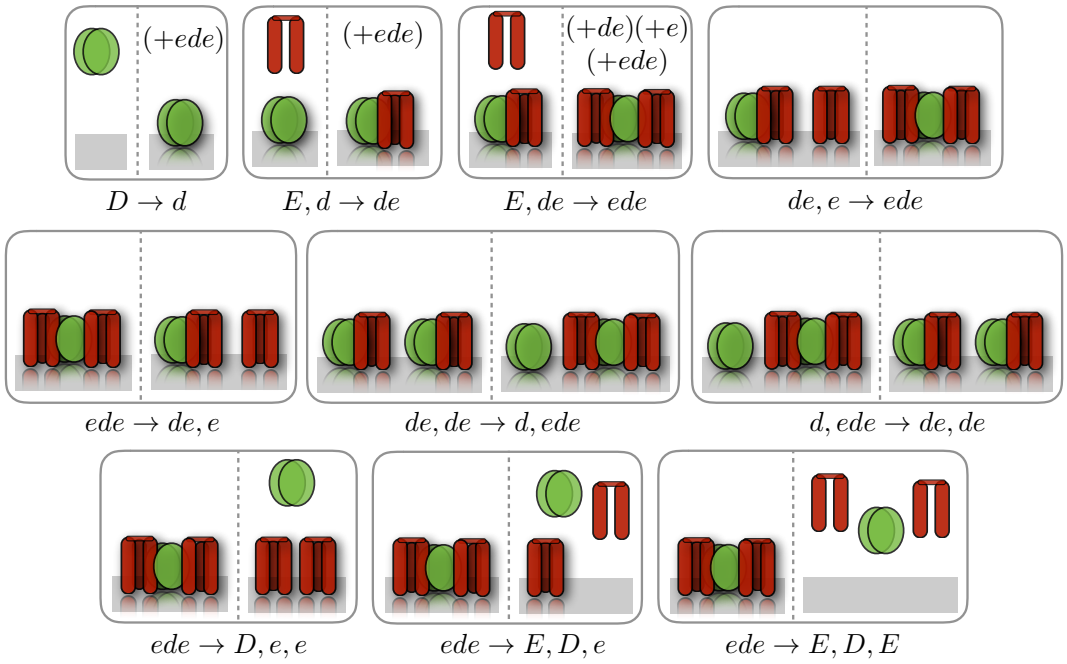
where $c_{\max}$ is the saturation concentration of MinD dimers on the membrane, $c_{\bar{d}}$ is the constant concentration of persistently bound MinD dimers on the membrane, an experimental artifact discussed in Section F.4.4, and $\omega_{u,v\to x,y}$ denotes the reaction rate of c_u and c_v converting into c_x and c_y, for $u, v, x, y \in \{\emptyset, D, E, d, de, ede, e\}$. When $\omega_{u,v\to x,y}$ has a superscript it indicates facilitation of the reaction by the superscripted species. I note that $\omega^{z}_{D\to d}$ has a multiplicative factor of c_D built into it for $z \in \{\emptyset, d, de, ede\}$ and $\omega^{z}_{E,d\to de}$ has a multiplicative factor of c_E built into it for $z \in \{\emptyset, de, ede, e\}$. The Symmetric Activation Model is depicted in Figure 4.9.

$\omega_{D\to d}$	$\omega^{d}_{D\to d}$	$\omega^{de}_{D\to d}$	$\bullet\omega^{ede}_{D\to d}$	$\omega_{E,d\to de}$	$\omega^{de}_{E,d\to de}$
$\omega^{e}_{E,d\to de}$	$\bullet\omega^{ede}_{E,d\to de}$	$\bullet\omega_{E,de\to ede}$	$\bullet\omega^{de}_{E,de\to ede}$	$\bullet\omega^{e}_{E,de\to ede}$	$\bullet\omega^{ede}_{E,de\to ede}$
$\omega_{d,e\to de}$	$\omega_{de\to d,e}$	$\bullet\omega_{de,e\to ede}$	$\bullet\omega_{ede\to de,e}$	$\omega_{d\to D}$	$\omega_{e\to E}$
$\bullet\omega_{de,de\to d,ede}$	$\bullet\omega_{d,ede\to de,de}$	$\bullet\omega_{ede\to D,e,e}$	$\bullet\omega_{ede\to E,D,e}$	$\bullet\omega_{ede\to E,D,E}$	
$\blacktriangle\omega_{de\to D,E}$	$\blacktriangle\omega_{de\to D,e}$				

(a)



(b)



(c)

Figure 4.9: The Symmetric Activation Model. Parameters characterizing reactions are shown in (a). A parameter that characterizes a reaction that is not included in the Extended Bonny Model is shown with a bullet (•) and the corresponding reaction is depicted in (c). A triangle (▲) next to a parameter indicates that the corresponding reaction from the Extended Bonny Model is not included in the Symmetric Activation Model. State variables are matched to protein states in (b). In (c), reactants are shown on the left and products are shown on the right of panels. (+) indicates facilitation in a reaction by the indicated species.

I fit the Symmetric Activation Model (4.7) to the near-homogeneous data, as described in Section 4.4. The resulting fit, state values, and parameter values are shown in Figure 4.10, Figure 4.11, and Table 4.4 respectively.

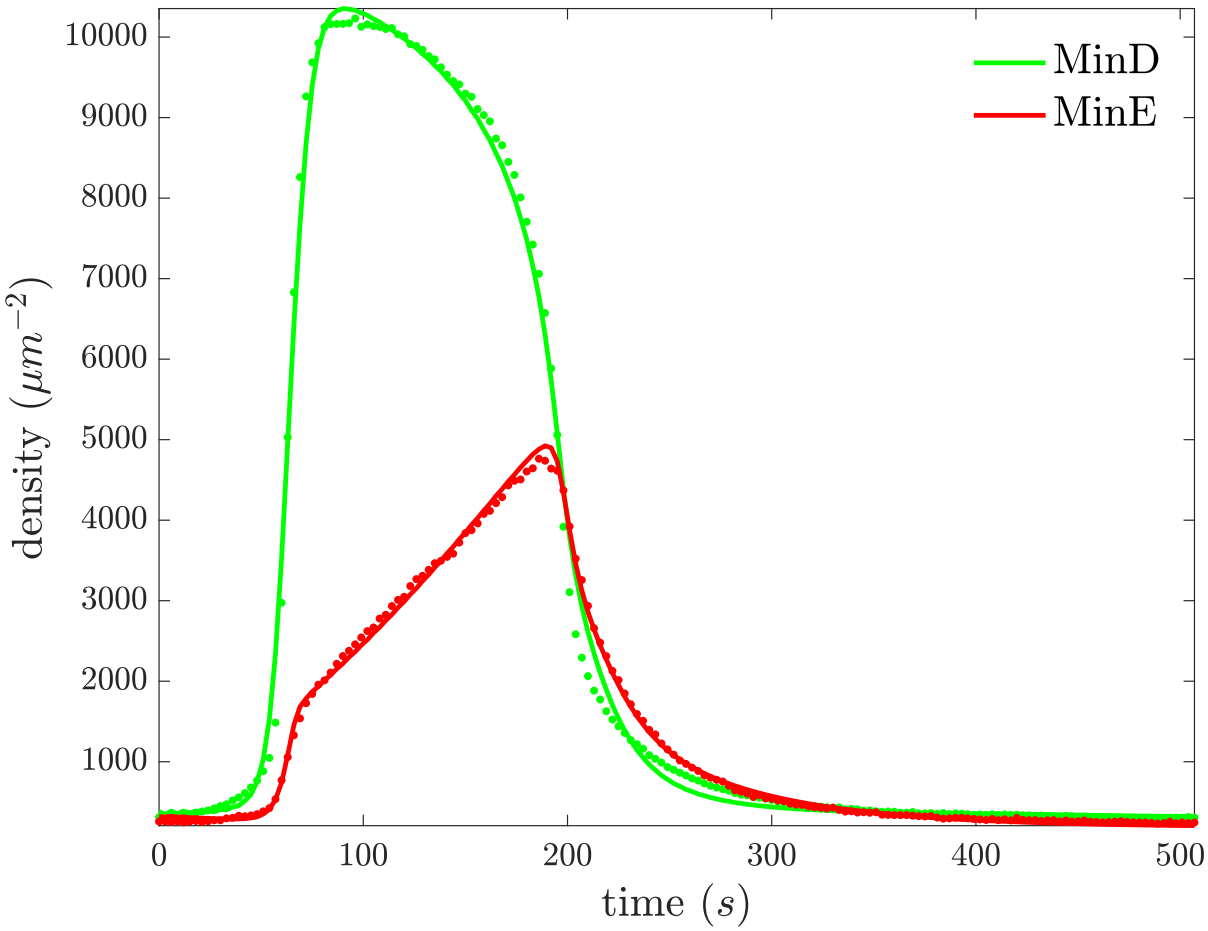


Figure 4.10: The fit of the Symmetric Activation Model to the near-homogeneous data. Data is shown with points and model values are shown with lines. The Symmetric Activation Model fits the near-homogeneous data moderately better than the Extended Bonny Model (compare to Figure 4.6).

Comparing Figures 4.6 and 4.10, the Symmetric Activation Model describes the near-homogeneous data visibly somewhat better than the Extended Bonny Model. Quantitatively, χ^2, the weighted sum of squared residuals, from the Extended Bonny Model is 1.89 times larger than χ^2 from the Symmetric Activation Model, and the value of AIC, the Akaike information criterion, from the Extended Bonny Model is 210 units larger than the value of AIC from the Symmetric Activation Model. Although the Symmetric Activation Model does describe the near-homogeneous data moderately better than the Extended Bonny Model, as discussed below, it does not agree well with experiments. Whereas, the Asymmetric Activation Model (described below) fits the the near-homogeneous data better than the Symmetric Activation Model and agrees well with experiments. Therefore, I do not expound the Symmetric Activation Model further.

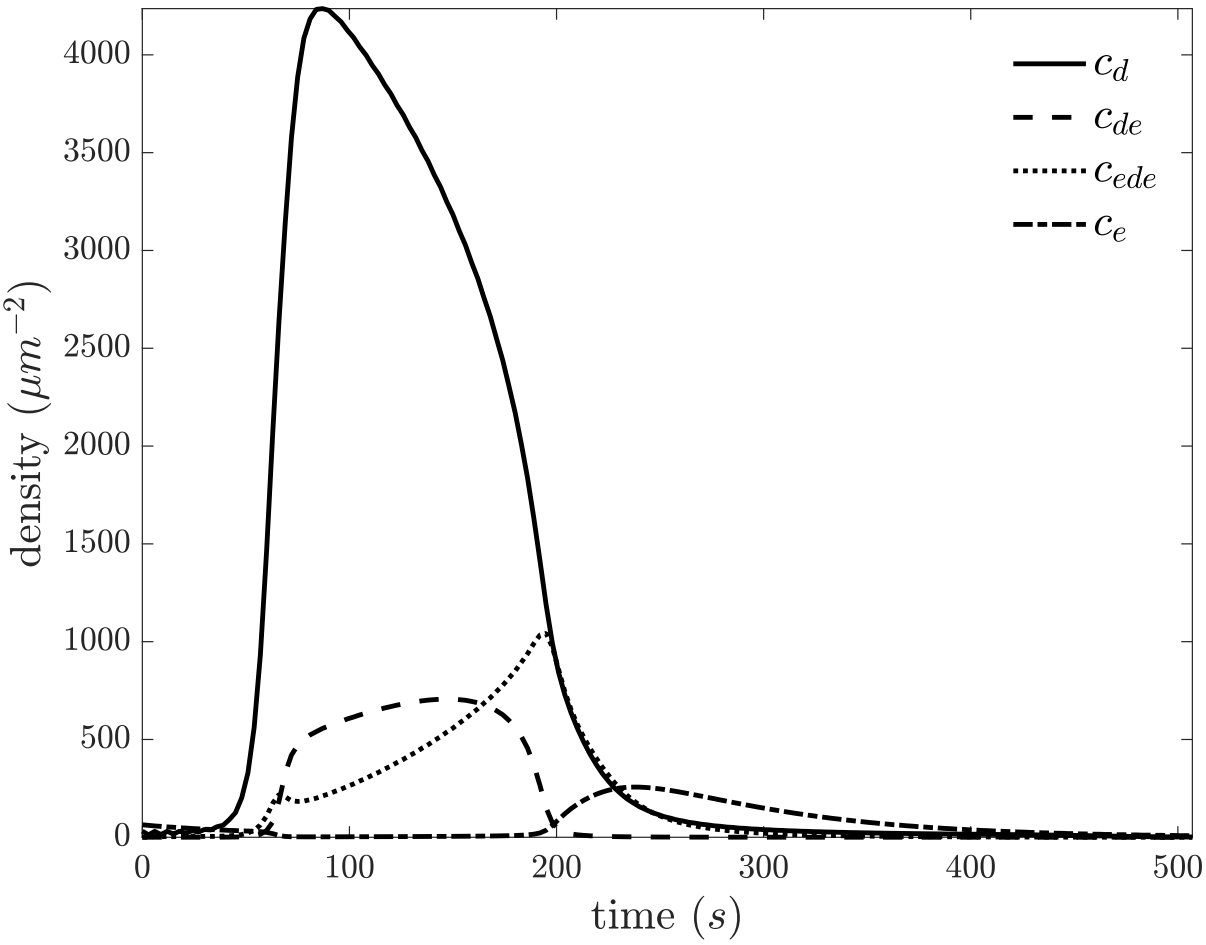


Figure 4.11: States from the fit of the Symmetric Activation Model to the near-homogeneous data.

parameter	value	95% confidence interval	units
C_d	$3.18 \cdot 10^2$	$[2.90 \cdot 10^2, 3.18 \cdot 10^2]$	μm^{-2}
C_e	$1.91 \cdot 10^2$	$[1.69 \cdot 10^2, 2.03 \cdot 10^2]$	μm^{-2}
$c_{\bar{d}}$	$0.00 \cdot 10^0$	$[0.00 \cdot 10^0, 7.69 \cdot 10^0]$	μm^{-2}
$c_{\max}$	$5.38 \cdot 10^3$	$[5.35 \cdot 10^3, 5.44 \cdot 10^3]$	μm^{-2}
c_s	$5.31 \cdot 10^1$	$[4.11 \cdot 10^1, 6.71 \cdot 10^1]$	μm^{-2}
$\omega_{D\to d}$	$0.00 \cdot 10^0$	$[0.00 \cdot 10^0, 2.37 \cdot 10^{-1}]$	$\mu m^{-2}\ s^{-1}$
$\omega^{d}_{D\to d}$	$2.62 \cdot 10^{-1}$	$[2.47 \cdot 10^{-1}, 2.68 \cdot 10^{-1}]$	s^{-1}
$\omega^{de}_{D\to d}$	$0.00 \cdot 10^0$	$[0.00 \cdot 10^0, 5.05 \cdot 10^{-2}]$	s^{-1}
$\omega^{ede}_{D\to d}$	$0.00 \cdot 10^0$	$[0.00 \cdot 10^0, 1.27 \cdot 10^{-2}]$	s^{-1}
$\omega_{E,d\to de}$	$1.01 \cdot 10^{-3}$	$[2.49 \cdot 10^{-4}, 1.29 \cdot 10^{-3}]$	s^{-1}
$\omega^{de}_{E,d\to de}$	$0.00 \cdot 10^0$	$[0.00 \cdot 10^0, 9.70 \cdot 10^{-7}]$	$\mu m^2\ s^{-1}$
$\omega^{e}_{E,d\to de}$	$8.17 \cdot 10^{-4}$	$[7.77 \cdot 10^{-4}, 8.61 \cdot 10^{-4}]$	$\mu m^2\ s^{-1}$
$\omega^{ede}_{E,d\to de}$	$0.00 \cdot 10^0$	$[0.00 \cdot 10^0, 1.00 \cdot 10^{-6}]$	$\mu m^2\ s^{-1}$
$\omega_{E,de\to ede}$	$0.00 \cdot 10^0$	$[0.00 \cdot 10^0, 5.13 \cdot 10^{-4}]$	s^{-1}
$\omega^{de}_{E,de\to ede}$	$0.00 \cdot 10^0$	$[0.00 \cdot 10^0, 2.80 \cdot 10^{-6}]$	$\mu m^2\ s^{-1}$
$\omega^{e}_{E,de\to ede}$	$8.11 \cdot 10^{-2}$	$[8.07 \cdot 10^{-2}, 8.16 \cdot 10^{-2}]$	$\mu m^2\ s^{-1}$
$\omega^{ede}_{E,de\to ede}$	$0.00 \cdot 10^0$	$[0.00 \cdot 10^0, 4.79 \cdot 10^{-6}]$	$\mu m^2\ s^{-1}$
$\omega_{d,e\to de}$	$0.00 \cdot 10^0$	$[0.00 \cdot 10^0, 9.04 \cdot 10^{-6}]$	$\mu m^2\ s^{-1}$
$\omega_{d,ede\to de,de}$	$1.69 \cdot 10^{-4}$	$[1.66 \cdot 10^{-4}, 1.70 \cdot 10^{-4}]$	$\mu m^2\ s^{-1}$
$\omega_{d\to D}$	$3.15 \cdot 10^{-1}$	$[2.93 \cdot 10^{-1}, 3.15 \cdot 10^{-1}]$	s^{-1}
$\omega_{de,de\to d,ede}$	$3.17 \cdot 10^{-4}$	$[3.13 \cdot 10^{-4}, 3.20 \cdot 10^{-4}]$	$\mu m^2\ s^{-1}$
$\omega_{de,e\to ede}$	$6.74 \cdot 10^{-3}$	$[6.38 \cdot 10^{-3}, 7.14 \cdot 10^{-3}]$	$\mu m^2\ s^{-1}$
$\omega_{de\to d,e}$	$0.00 \cdot 10^0$	$[0.00 \cdot 10^0, 1.55 \cdot 10^{-4}]$	s^{-1}
$\omega_{e\to E}$	$1.55 \cdot 10^{-2}$	$[1.36 \cdot 10^{-2}, 1.66 \cdot 10^{-2}]$	s^{-1}
$\omega_{ede\to D,e,e}$	$1.67 \cdot 10^{-4}$	$[0.00 \cdot 10^0, 5.56 \cdot 10^{-4}]$	s^{-1}
$\omega_{ede\to E,D,E}$	$2.29 \cdot 10^{-1}$	$[2.26 \cdot 10^{-1}, 2.31 \cdot 10^{-1}]$	s^{-1}
$\omega_{ede\to E,D,e}$	$4.12 \cdot 10^{-2}$	$[3.93 \cdot 10^{-2}, 4.20 \cdot 10^{-2}]$	s^{-1}
$\omega_{ede\to de,e}$	$3.08 \cdot 10^{-7}$	$[0.00 \cdot 10^0, 9.00 \cdot 10^{-4}]$	s^{-1}

Table 4.4: Parameters from the fit of the Symmetric Activation Model to the near-homogeneous data. C_d and C_e are as described in Table 4.1. Details of calculating confidence intervals are described in Section 4.4.5.

The Asymmetric Activation Model

Apart from a crystal structure showing two MinE fragments (containing 20 of 88 amino acids) each bound to a subunit of the MinD dimer [52], there is no direct evidence for the Symmetric Activation Model. Contrarily, a MinE dimer bound to one subunit of a MinD dimer stimulates ATPase activity in both subunits of the MinD dimer [53], showing asymmetric rather than symmetric activation of MinD by MinE. Furthermore, with ATP$_\gamma$S, a non-hydrolyzable analogue of ATP, and a high concentration of MinE, the Symmetric Activation Model would predict that the ratio of MinD to MinE on a lipid bilayer would be 1:2. However, experiments testing precisely this scenario found a ratio of MinD to MinE of 1:1 [34] and 3:1 [42]. Additionally, with ATP$_\gamma$S, MinD and MinE dissociate with a ratio of 1:1 from the supported lipid bilayer

[73], suggesting that, if dissociation occurs predominantly in the unstable MinD-MinE complex, the ratio of MinD to MinE in the unstable MinD-MinE complex is 1:1. Therefore, experimental outcomes support asymmetric rather than symmetric activation of MinD by MinE.

For asymmetric activation of MinD by MinE, MinD and MinE form an unstable complex with one MinE dimer bound to one subunit of the MinD dimer ([52], [53]). There is no direct evidence for the structure of a stable MinD-MinE complex. However, a crystal structure shows a MinE dimer bridging two MinD dimers, with one MinD dimer rotated 90° with respect to the other MinD dimer [52]. Because of the 90° rotation of one MinD dimer with respect to the other MinD dimer, a MinE dimer bridging two MinD dimers has been considered more of an experimental artifact than a biologically relevant state. I believe, however, that a MinE dimer bridging two MinD dimers may reflect the stable MinD-MinE complex, and the 90° rotation of one MinD dimer with respect to the other MinD dimer may reflect the stabilization mechanism. If the stable configuration of a MinE dimer bridging two MinD dimers requires a 90° rotation of one MinD dimer with respect to the other MinD dimer, then some strain would likely exist within a MinE dimer that bridges two membrane-bound MinD dimers both with membrane-targeting sequences oriented toward the membrane. I hypothesize that when a MinE dimer binds to a second MinD dimer on the membrane, imposed strain alters the interaction between the MinE dimer and the first MinD dimer, tempering MinE-stimulated ATPase activity. Thus, I propose that MinD and MinE form a stable complex with one MinE dimer bridging two MinD dimers.

Supporting my stabilization-by-bridging hypothesis, experiments show that removing the dimerization domain of MinE removes switch-like behavior in the stimulation of MinD ATPase activity by MinE: the rate of MinD ATPase activity as a function of WT MinE concentration resembles a Hill equation with a Hill coefficient greater than one, while the rate of MinD ATPase activity as a function of dimerization-domain-deficient MinE concentration resembles a Hill equation with a Hill coefficient of one [73]. Refuting my stabilization-by-bridging hypothesis, dimerization-domain-deficient MinE stabilizes MinD membrane binding in buffer-flow experiments [73]. However, it stabilizes MinD membrane binding less than WT MinE. Also, dimerization-domain-deficient MinE does not support dynamic pattern formation *in vitro*, and the rate of MinD ATPase activity is lower with high concentrations of dimerization-domain-deficient MinE than with high concentrations of WT MinE [73]. So, the stabilization of MinD membrane binding could stem from a dimerization-domain-deficient disruption of the regular MinD-MinE interaction, which tempers the stimulation of ATPase activity.

I build on the Extended Bonny Model (4.6) to develop the Asymmetric Activation Model. My Asymmetric Activation Model consists of four supported-lipid-bilayer-bound states, c_d, c_{de}, c_{ded}, and c_e, respectively corresponding to the supported-lipid-bilayer-bound concentrations of MinD dimers, MinE dimers with one MinD dimer bound, MinE dimers with two MinD dimers bound, and MinE dimers with no MinD dimers bound. For asymmetric activation, c_{de} is the unstable state and c_{ded} is the stable state. Thus, I include reactions in the Asymmetric

Activation Model where c_{de} dissociates from the supported lipid bilayer, and I exclude reactions from the Asymmetric Activation Model where c_{ded} dissociates from the supported lipid bilayer. As stated previously, a MinE dimer bound to one subunit of a MinD dimer is able to cause a conformational change in the other subunit of the MinD dimer [53], and, with $ATP_{\gamma}S$ and a high concentration of MinE, the number of MinE dimers does not exceed the number of MinD dimers on lipid bilayers ([34], [42]). Thus, in the Asymmetric Activation Model, I assume that the binding of a MinE dimer to one subunit of a MinD dimer excludes other MinE dimers from binding to the other subunit of the MinD dimer. As such, without reason for restriction, I allow MinE-exclusion reactions in the Asymmetric Activation Model:

$$c_E + c_{ded} \rightarrow 2c_{de}, \tag{4.8a}$$

$$c_e + c_{ded} \rightarrow 2c_{de}, \tag{4.8b}$$

$$c_{de} + c_{de} \rightarrow c_{ded} + c_e, \tag{4.8c}$$

where c_E is the concentration of bulk MinE dimers. Also, without reason for restriction, in the Asymmetric Activation Model, I allow c_d, c_{de}, and c_{ded} to recruit c_D, the concentration of bulk MinD dimers, to the supported lipid bilayer, and I allow c_{de}, c_{ded}, and c_e to recruit c_E to bind to c_d and c_{ded}. As in the Extended Bonny Model, I only consider reactions where MinE does not spontaneously dissociate from MinD on the supported lipid bilayer. Thus, in the Asymmetric Activation Model, I include forward and reverse bimolecular reactions of c_d, c_{de}, c_{ded}, and c_e with all products in $\{c_d, c_{de}, c_{ded}, c_e\}$. Also, as in the Extended Bonny Model, in the Asymmetric Activation Model, I include the spontaneous dissociation of c_d with stabilization by

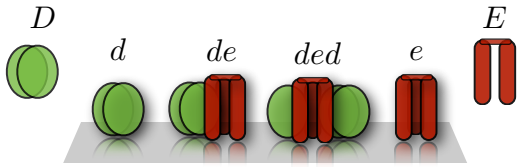
supported-lipid-bilayer-bound MinD dimers. I define the Asymmetric Activation Model:

$$\begin{aligned}
\frac{dc_d}{dt} &= (\omega_{D\to d} + \omega^{d}_{D\to d}c_d + \omega^{de}_{D\to d}c_{de} + \omega^{ded}_{D\to d}c_{ded})(c_{\max} - c_{\bar{d}} - c_d - c_{de} - 2c_{ded})/c_{\max} \\
&\quad - (\omega_{E,d\to de} + \omega^{de}_{E,d\to de}c_{de} + \omega^{ded}_{E,d\to de}c_{ded} + \omega^{e}_{E,d\to de}c_e)c_d \\
&\quad - \omega_{d,de\to ded}c_d c_{de} + \omega_{ded\to d,de}c_{ded} - \omega_{d,e\to de}c_d c_e + \omega_{de\to d,e}c_{de} \\
&\quad - \omega_{d\to D}c_s c_d/(c_s + c_{\bar{d}} + c_d + c_{de} + 2c_{ded}), \qquad (4.9a) \\
\frac{dc_{de}}{dt} &= (\omega_{E,d\to de} + \omega^{de}_{E,d\to de}c_{de} + \omega^{ded}_{E,d\to de}c_{ded} + \omega^{e}_{E,d\to de}c_e)c_d \\
&\quad + 2(\omega_{E,ded\to de,de} + \omega^{de}_{E,ded\to de,de}c_{de} + \omega^{ded}_{E,ded\to de,de}c_{ded} + \omega^{e}_{E,ded\to de,de}c_e)c_{ded} \\
&\quad - \omega_{d,de\to ded}c_d c_{de} + \omega_{ded\to d,de}c_{ded} + \omega_{d,e\to de}c_d c_e - \omega_{de\to d,e}c_{de} \\
&\quad - 2\omega_{de,de\to ded,e}c^2_{de} + 2\omega_{ded,e\to de,de}c_{ded}c_e - \omega_{de\to D,E}c_{de} - \omega_{de\to D,e}c_{de}, \qquad (4.9b) \\
\frac{dc_{ded}}{dt} &= -(\omega_{E,ded\to de,de} + \omega^{de}_{E,ded\to de,de}c_{de} + \omega^{ded}_{E,ded\to de,de}c_{ded} + \omega^{e}_{E,ded\to de,de}c_e)c_{ded} \\
&\quad + \omega_{d,de\to ded}c_d c_{de} - \omega_{ded\to d,de}c_{ded} + \omega_{de,de\to ded,e}c^2_{de} - \omega_{ded,e\to de,de}c_{ded}c_e, \qquad (4.9c) \\
\frac{dc_e}{dt} &= \omega_{de\to d,e}c_{de} - \omega_{d,e\to de}c_d c_e + \omega_{de,de\to ded,e}c^2_{de} - \omega_{ded,e\to de,de}c_{ded}c_e \\
&\quad + \omega_{de\to D,e}c_{de} - \omega_{e\to E}c_e, \qquad (4.9d)
\end{aligned}$$

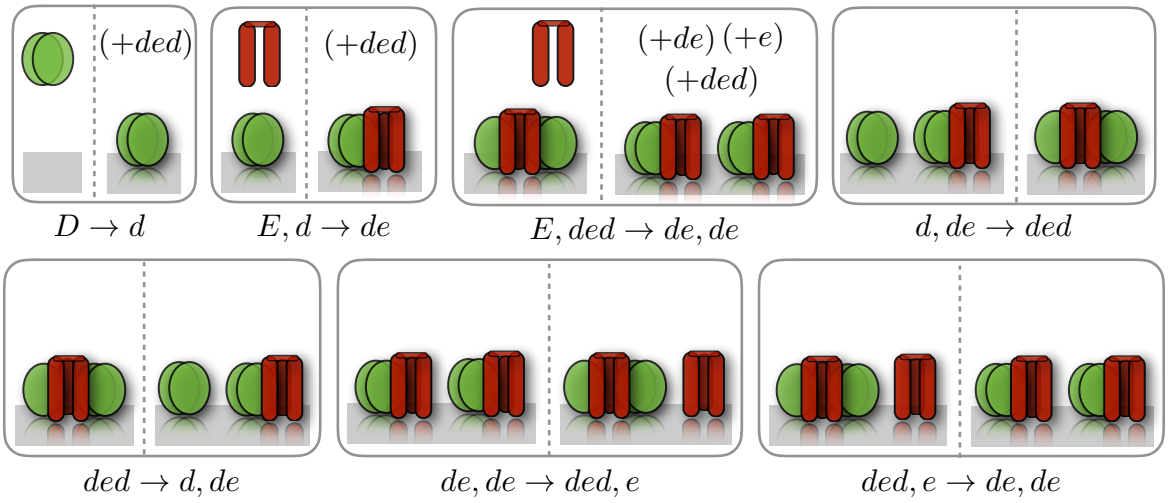
where $c_{\max}$ is the saturation concentration of MinD dimers on the membrane, $c_{\bar{d}}$ is the constant concentration of persistently bound MinD dimers on the membrane, an experimental artifact discussed in Section F.4.4, and $\omega_{u,v\to x,y}$ denotes the reaction rate of c_u and c_v converting into c_x and c_y, for $u, v, x, y \in \{\emptyset, D, E, d, de, ded, e\}$. When $\omega_{u,v\to x,y}$ has a superscript it indicates facilitation of the reaction by the superscripted species. I note that $\omega^{z}_{D\to d}$ has a multiplicative factor of c_D built into it for $z \in \{\emptyset, d, de, ded\}$ and $\omega^{z}_{E,d\to de}$ has a multiplicative factor of c_E built into it for $z \in \{\emptyset, de, ded, e\}$. The Asymmetric Activation Model is depicted in Figure 4.12.

$\omega_{D\to d}$	$\omega^{d}_{D\to d}$	$\omega^{de}_{D\to d}$	$\bullet\omega^{ded}_{D\to d}$	$\omega_{E,d\to de}$
$\omega^{de}_{E,d\to de}$	$\bullet\omega^{ded}_{E,d\to de}$	$\omega^{e}_{E,d\to de}$	$\bullet\omega_{E,ded\to de,de}$	$\bullet\omega^{de}_{E,ded\to de,de}$
$\bullet\omega^{e}_{E,ded\to de,de}$	$\bullet\omega^{ded}_{E,ded\to de,de}$	$\omega_{d,e\to de}$	$\omega_{de\to d,e}$	$\bullet\omega_{d,de\to ded}$
$\bullet\omega_{ded\to d,de}$	$\omega_{de\to D,E}$	$\omega_{de\to D,e}$	$\bullet\omega_{de,de\to ded,e}$	$\bullet\omega_{ded,e\to de,de}$
$\omega_{d\to D}$	$\omega_{e\to E}$			

(a)



(b)



(c)

Figure 4.12: The Asymmetric Activation Model. Parameters characterizing reactions are shown in (a). A parameter that characterizes a reaction that is not included in the Extended Bonny Model is shown with a bullet (•) and the corresponding reaction is depicted in (c). All reactions from the Extended Bonny Model are included in the Asymmetric Activation Model. State variables are matched to protein states in (b). In (c), reactants are shown on the left and products are shown on the right of panels. (+) indicates facilitation in a reaction by the indicated species.

I fit the Asymmetric Activation Model (4.9) to the near-homogeneous data, as described in Section 4.4. The resulting fit, state values, and parameter values are shown in Figure 4.13, Figure 4.14, and Table 4.5 respectively.

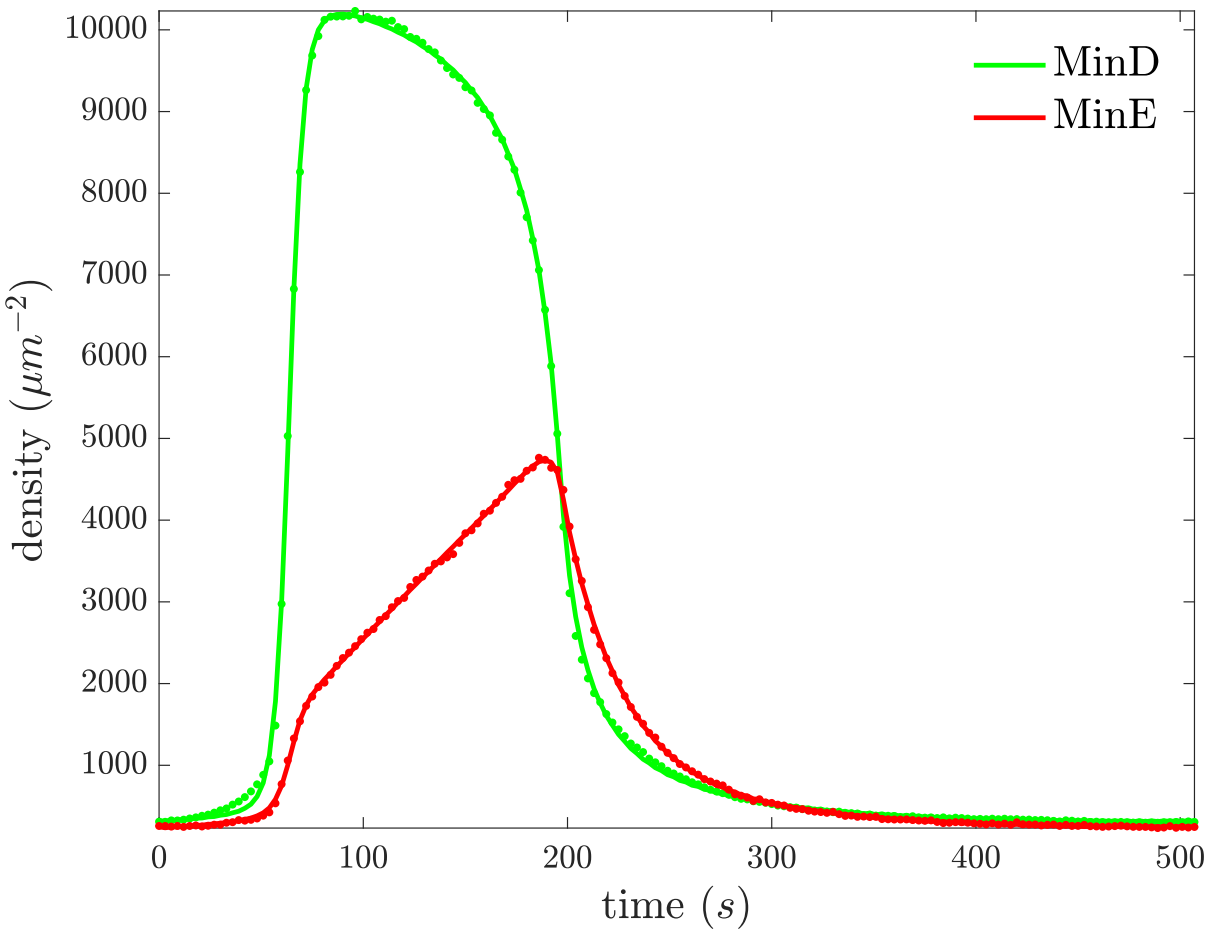


Figure 4.13: The fit of the Asymmetric Activation Model to the near-homogeneous data. Data is shown with points and model values are shown with lines. The Asymmetric Activation Model fits the near-homogeneous data appreciably better than the Symmetric Activation Model (compare to Figure 4.10).

Comparing Figures 4.6, 4.10, and 4.13, the Asymmetric Activation Model describes the near-homogeneous data visibly better than the Extended Bonny Model, even more so than the Symmetric Activation Model. Quantitatively, χ^2, the weighted sum of squared residuals, from the Extended Bonny Model is 16.1 times larger than χ^2 from the Asymmetric Activation Model, and the value of AIC, the Akaike information criterion, from the Extended Bonny Model is 919 units larger than the value of AIC from the Asymmetric Activation Model. Comparatively, χ^2 from the Extended Bonny Model is 1.89 times larger than χ^2 from the Symmetric Activation Model, and the value of AIC from the Extended Bonny Model is 210 units larger than the value of AIC from the Symmetric Activation Model. As such, the Asymmetric Activation Model describes the near-homogeneous data considerably better than the Symmetric Activation Model.

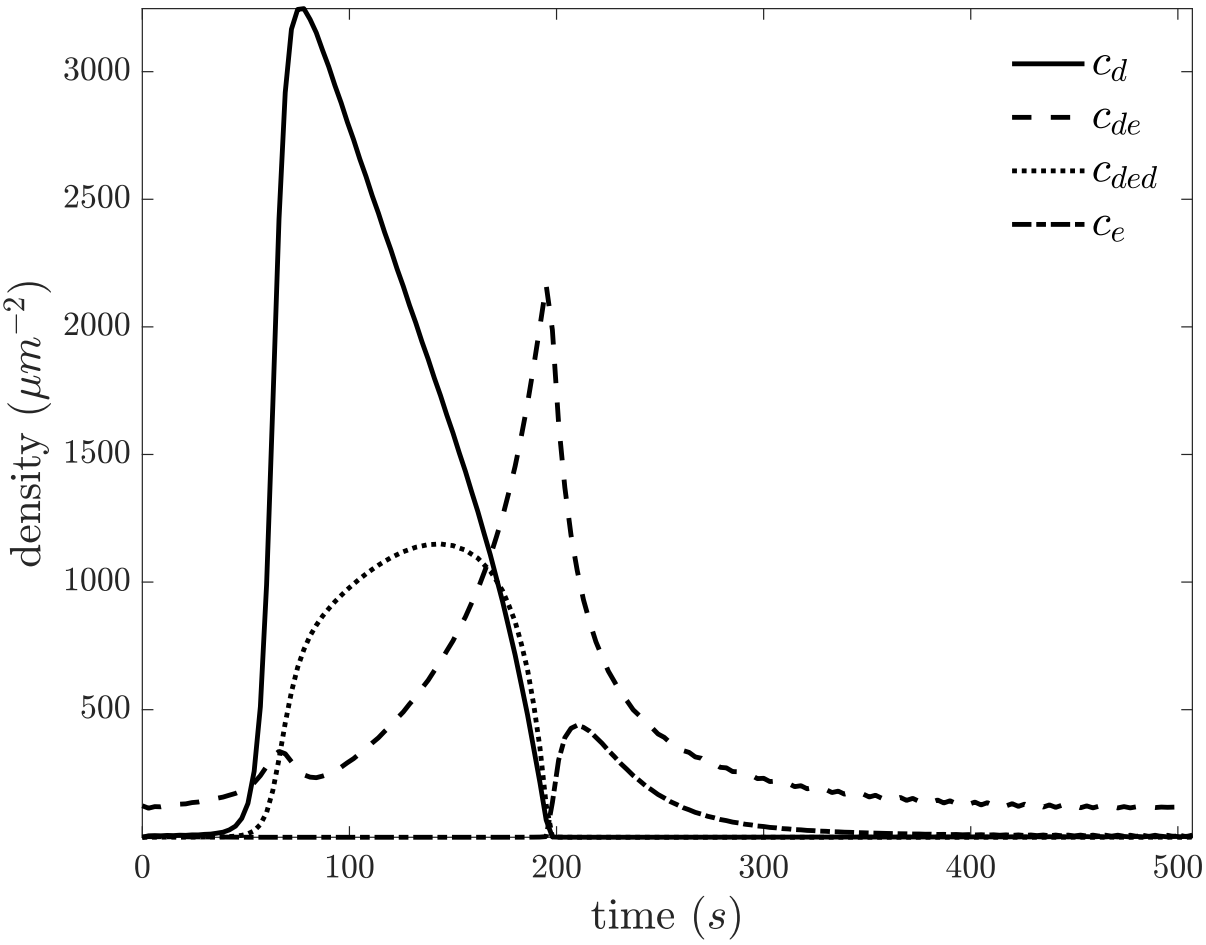


Figure 4.14: States from the fit of the Asymmetric Activation Model to the near-homogeneous data.

parameter	value	95% confidence interval	units
C_d	$7.47 \cdot 10^{1}$	$[6.52 \cdot 10^{1}, 8.58 \cdot 10^{1}]$	μm^{-2}
C_e	$6.23 \cdot 10^{-1}$	$[0.00 \cdot 10^{0}, 6.67 \cdot 10^{0}]$	μm^{-2}
$c_{\bar{d}}$	$1.08 \cdot 10^{-2}$	$[0.00 \cdot 10^{0}, 1.42 \cdot 10^{1}]$	μm^{-2}
$c_{\max}$	$5.30 \cdot 10^{3}$	$[5.28 \cdot 10^{3}, 5.35 \cdot 10^{3}]$	μm^{-2}
c_s	$1.15 \cdot 10^{2}$	$[8.11 \cdot 10^{1}, 2.64 \cdot 10^{2}]$	μm^{-2}
$\omega_{D\to d}$	$8.77 \cdot 10^{-5}$	$[0.00 \cdot 10^{0}, 9.66 \cdot 10^{-2}]$	$\mu m^{-2}\ s^{-1}$
$\omega^{d}_{D\to d}$	$3.73 \cdot 10^{-1}$	$[3.58 \cdot 10^{-1}, 3.96 \cdot 10^{-1}]$	s^{-1}
$\omega^{de}_{D\to d}$	$2.00 \cdot 10^{-1}$	$[1.98 \cdot 10^{-1}, 2.02 \cdot 10^{-1}]$	s^{-1}
$\omega^{ded}_{D\to d}$	$4.20 \cdot 10^{-1}$	$[4.05 \cdot 10^{-1}, 4.29 \cdot 10^{-1}]$	s^{-1}
$\omega_{E,d\to de}$	$3.28 \cdot 10^{-3}$	$[2.34 \cdot 10^{-3}, 3.42 \cdot 10^{-3}]$	s^{-1}
$\omega^{de}_{E,d\to de}$	$2.11 \cdot 10^{-9}$	$[0.00 \cdot 10^{0}, 2.41 \cdot 10^{-6}]$	$\mu m^{2}\ s^{-1}$
$\omega^{ded}_{E,d\to de}$	$8.50 \cdot 10^{-10}$	$[0.00 \cdot 10^{0}, 8.21 \cdot 10^{-7}]$	$\mu m^{2}\ s^{-1}$
$\omega^{e}_{E,d\to de}$	$1.90 \cdot 10^{0}$	$[1.88 \cdot 10^{0}, 1.92 \cdot 10^{0}]$	$\mu m^{2}\ s^{-1}$
$\omega_{E,ded\to de,de}$	$1.86 \cdot 10^{-13}$	$[0.00 \cdot 10^{0}, 7.57 \cdot 10^{-4}]$	s^{-1}
$\omega^{de}_{E,ded\to de,de}$	$1.45 \cdot 10^{-5}$	$[1.35 \cdot 10^{-5}, 1.50 \cdot 10^{-5}]$	$\mu m^{2}\ s^{-1}$
$\omega^{ded}_{E,ded\to de,de}$	$2.10 \cdot 10^{-5}$	$[1.97 \cdot 10^{-5}, 2.11 \cdot 10^{-5}]$	$\mu m^{2}\ s^{-1}$
$\omega^{e}_{E,ded\to de,de}$	$1.64 \cdot 10^{-1}$	$[1.63 \cdot 10^{-1}, 1.65 \cdot 10^{-1}]$	$\mu m^{2}\ s^{-1}$
$\omega_{d,de\to ded}$	$4.86 \cdot 10^{-5}$	$[4.75 \cdot 10^{-5}, 5.00 \cdot 10^{-5}]$	$\mu m^{2}\ s^{-1}$
$\omega_{d,e\to de}$	$6.16 \cdot 10^{-1}$	$[6.11 \cdot 10^{-1}, 6.22 \cdot 10^{-1}]$	$\mu m^{2}\ s^{-1}$
$\omega_{d\to D}$	$3.15 \cdot 10^{-1}$	$[2.25 \cdot 10^{-1}, 3.15 \cdot 10^{-1}]$	s^{-1}
$\omega_{de,de\to ded,e}$	$6.59 \cdot 10^{-4}$	$[6.59 \cdot 10^{-4}, 6.59 \cdot 10^{-4}]$	$\mu m^{2}\ s^{-1}$
$\omega_{de\to D,E}$	$2.00 \cdot 10^{-1}$	$[1.99 \cdot 10^{-1}, 2.02 \cdot 10^{-1}]$	s^{-1}
$\omega_{de\to D,e}$	$3.40 \cdot 10^{-6}$	$[0.00 \cdot 10^{0}, 5.66 \cdot 10^{-4}]$	s^{-1}
$\omega_{de\to d,e}$	$6.64 \cdot 10^{-2}$	$[6.59 \cdot 10^{-2}, 6.68 \cdot 10^{-2}]$	s^{-1}
$\omega_{ded,e\to de,de}$	$9.34 \cdot 10^{0}$	$[9.34 \cdot 10^{0}, 9.34 \cdot 10^{0}]$	$\mu m^{2}\ s^{-1}$
$\omega_{ded\to d,de}$	$1.52 \cdot 10^{-13}$	$[0.00 \cdot 10^{0}, 1.91 \cdot 10^{-3}]$	s^{-1}
$\omega_{e\to E}$	$5.39 \cdot 10^{-2}$	$[5.34 \cdot 10^{-2}, 5.45 \cdot 10^{-2}]$	s^{-1}

Table 4.5: Parameters from the fit of the Asymmetric Activation Model to the near-homogeneous data. C_d and C_e are as described in Table 4.1. Details of calculating confidence intervals are described in Section 4.4.5.

To determine how individual reactions affect the Asymmetric Activation Model's ability to describe the near-homogeneous data, individually, I remove non-necessary reactions from the Asymmetric Activation Model and fit the resulting model to the near-homogeneous data, as described in Section 4.4. Results are shown in Table 4.6.

Null parameter	$\chi^2/\chi^2_\emptyset$	$\text{AIC} - \text{AIC}_\emptyset$
$\omega^{d}_{D\to d}$	3.80	$4.36 \cdot 10^{2}$
$\omega^{de}_{D\to d}$	5.71	$5.82 \cdot 10^{2}$
$\omega^{ded}_{D\to d}$	1.98	$2.30 \cdot 10^{2}$
$\omega^{de}_{E,d\to de}$	1.00	$-4.34 \cdot 10^{0}$
$\omega^{ded}_{E,d\to de}$	1.05	$7.16 \cdot 10^{1}$
$\omega^{e}_{E,d\to de}$	8.43	$7.09 \cdot 10^{2}$
$\omega_{E,ded\to de,de}$	1.00	$-7.20 \cdot 10^{-1}$
$\omega^{de}_{E,ded\to de,de}$	1.07	$2.09 \cdot 10^{1}$
$\omega^{ded}_{E,ded\to de,de}$	1.12	$3.35 \cdot 10^{1}$
$\omega^{e}_{E,ded\to de,de}$	1.13	$3.48 \cdot 10^{1}$
$\omega_{d,de\to ded}$	4.46	$4.93 \cdot 10^{2}$
$\omega_{d,e\to de}$	1.49	$1.23 \cdot 10^{2}$
$\omega_{d\to D}$	1.04	$4.56 \cdot 10^{0}$
$\omega_{de,de\to ded,e}$	2.11	$2.43 \cdot 10^{2}$
$\omega_{de\to D,E}$	5.81	$5.82 \cdot 10^{2}$
$\omega_{de\to D,e}$	1.04	$1.13 \cdot 10^{1}$
$\omega_{de\to d,e}$	1.05	$1.33 \cdot 10^{1}$
$\omega_{ded,e\to de,de}$	4.71	$5.09 \cdot 10^{2}$
$\omega_{ded\to d,de}$	1.01	$2.29 \cdot 10^{0}$

Table 4.6: Removed-reaction fits of the Asymmetric Activation Model to the near-homogeneous data. A removed reaction is characterized by a null parameter. χ^2 is the the weighted sum of squared residuals, and AIC is the Akaike information criterion. $\chi^2_\emptyset$ and $\text{AIC}_\emptyset$ are the values of χ^2 and AIC from the Asymmetric Activation Model without a removed reaction. $\chi^2/\chi^2_\emptyset$ and $\text{AIC} - \text{AIC}_\emptyset$ measure the affect of removing the reaction characterized by the null parameter from the Asymmetric Activation Model on its ability to fit the near-homogeneous data; larger values of $\chi^2/\chi^2_\emptyset$ and $\text{AIC} - \text{AIC}_\emptyset$ correspond to a larger decrease in fitting ability.

As shown in Table 4.3, the reactions characterized by the parameters $\omega^{d}_{D\to d}$, $\omega^{de}_{D\to d}$, $\omega^{ded}_{D\to d}$, $\omega^{e}_{E,d\to de}$, $\omega_{d,de\to ded}$, $\omega_{d,e\to de}$, $\omega_{de,de\to ded,e}$, $\omega_{de\to D,E}$, and $\omega_{ded,e\to de,de}$ each significantly ($\text{AIC} - \text{AIC}_\emptyset > 50$) affect the Asymmetric Activation Model's ability to describe the near-homogeneous data.

As is visible in Figure 4.14 and elaborated in Figure 4.15, at the front end of the pulse, a large proportion of MinD dimers are in the semistable state, c_d, or in the stable state, c_{ded}. Whereas, at the back end of the pulse, a large proportion of MinD dimers are in the unstable state, c_{de}.

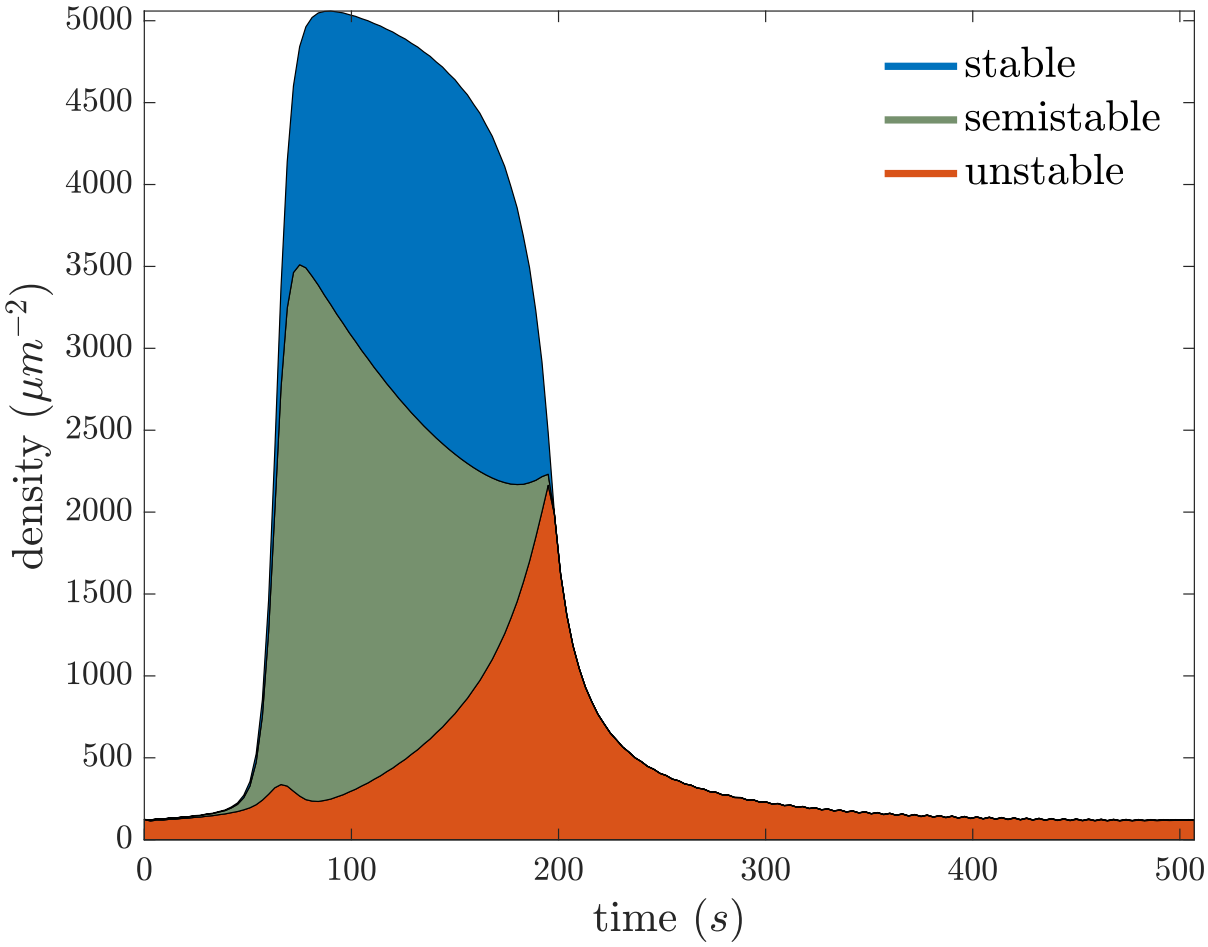


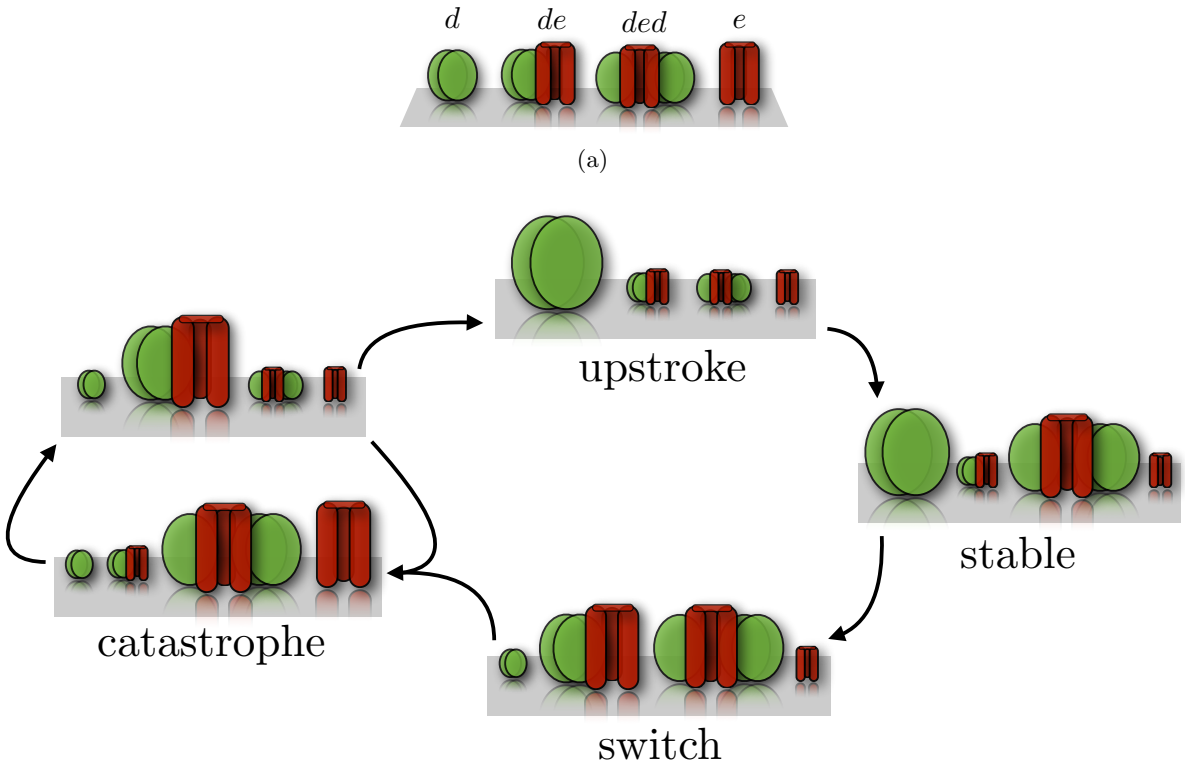
Figure 4.15: Stability of MinD dimers on the supported lipid bilayer. The total MinD dimer concentration on the supported lipid bilayer is separated into the concentration of dimers in the stable state, $2c_{ded}$, the concentration of dimers in the semistable state, c_d, and the concentration of dimers in the unstable state, c_{de}. Values of c_d, c_{de}, and c_{ded}, as shown in Figure 4.14, come from the fit of the Asymmetric Activation Model to the near-homogeneous data. MinD dimers are predominantly semi-stably and stably bound to the supported lipid bilayer at the front end of the pulse and unstably bound to the supported lipid bilayer at the back end of the pulse.

4.3.4 A Stability-Switching Mechanism Underlying the Dynamic Behavior of the Min System

Through the fit of the Asymmetric Activation Model to the near-homogeneous data, I interpret a stability-switching mechanism that underlies the dynamic behavior of the Min system. My discussion is based on the state values shown in Figure 4.14 and the characterization of significant reactions from Table 4.6.

During the MinD upstroke, bulk MinD binds to the supported lipid bilayer in the protein state d, which is semi-stably bound to the supported lipid bilayer, and recruits more bulk MinD to the supported lipid bilayer through the significant reaction $D + d \rightarrow 2d$ (characterized by

$\omega^{d}_{D\to d}$). Bulk MinE binds to supported-lipid-bilayer-bound MinD in the protein state de, which is unstably bound to the supported lipid bilayer. However, through the significant reaction $d+de \to ded$ (characterized by $\omega_{d,de\to ded}$), a large concentration of d pushes de to react with d to form ded, the protein state that is stably bound to the supported lipid bilayer. Simultaneously, through the significant reaction $d + e \to de$ (characterized by $\omega_{d,e\to de}$), a large concentration of d pushes the protein state e to react with d to form de, which a large concentration of d pushes to react with d to form ded. As more bulk MinE binds to supported-lipid-bilayer-bound MinD, the concentration of d decreases and the concentration of de increases. Through the significant reaction $de + de \to ded + e$ (characterized by $\omega_{de,de\to ded,e}$) and the reactions $de \to D + e$ and $de \to d + e$ (characterized by $\omega_{de\to D,e}$ and $\omega_{de\to d,e}$), de generates e. Eventually, in catastrophe, a period of destabilization with positive feedback, without d as a substrate, e binds to ded and generates de through the significant reaction $ded + e \to de + de$ (characterized by $\omega_{ded,e\to de,de}$), de generates more e which generates more de, and de dissociates from the supported lipid bilayer through the significant reaction $de \to D + E$ (characterized by $\omega_{de\to D,E}$). Capping catastrophe, e recruits bulk MinE to bind to any remaining d through the significant reaction $E+d+e \to de+e$ (characterized by $\omega^{e}_{E,d\to de}$), and e dissociates from the supported lipid bilayer through the reaction $e \to E$ (characterized by $\omega_{e\to E}$). Collectively, a large concentration of d reinforces stability of MinD on the supported lipid bilayer and suppresses the catastrophe switch, e; a decrease in the concentration of d signals the switch from stability to catastrophe, causing rapid MinD dissociation from the supported lipid bilayer. The aforementioned stability-switching mechanism is depicted in Figure 4.16.



(a)

(b)

Figure 4.16: The stability-switching mechanism. Protein states are shown in (a). The stability-switching mechanism is shown in (b). Relative concentration/affect is shown by size. The semistable protein state d is predominant during pulse upstroke (top). Bulk MinE binds to d to form the unstable protein state de, but a large concentration of d pushes de to react with d to form the stable protein state ded (right). Meanwhile, a large concentration of d inhibits the catastrophe switch, protein state e, by reacting with e to form de. The concentration of d decreases and the switch in stability occurs (bottom). In a feedback loop, de generates e, the catastrophe switch, which binds to ded to form more de (left); Min proteins rapidly dissociate from the supported lipid bilayer in catastrophe.

4.3.5 Results Relating to Experimental Observations

The results from my modeling and fitting could explain some experimental observations. A MinE mutant that stimulates MinD ATPase activity but is deficient in membrane binding, MinE C1 [27], fails to stimulate dynamic pattern formation on a supported lipid bilayer *in vitro* [44]. Rather, MinD and MinE C1 form a stationary structure on the supported lipid bilayer with MinD and MinE C1 profiles that are similar in shape to MinD and MinE profiles in traveling waves, except that the MinE C1 profile lacks a sharp peak [44]. My results, from fitting both the Extended Bonny Model and the Asymmetric Activation Model to the near-homogeneous data, suggest that supported-lipid-bilayer-bound MinE recruits bulk MinE to bind to supported-lipid-bilayer-bound MinD ($\omega^e_{E,d\to de}$ is significant). As such, the lack of a sharp peak in the

MinE C1 profile could follow from the inability of MinE C1 to recruit bulk MinE C1 to bind to supported-lipid-bilayer-bound MinD. My results, from fitting the Asymmetric Activation Model to the near-homogeneous data, also suggest that supported-lipid-bilayer-bound MinE acts as a catastrophe switch ($\omega_{ded,e\to de,de}$ is significant). Thus, MinD and MinE C1 forming a stable, stationary structure on the supported lipid bilayer could follow from the lack of a catastrophe switch with the MinE C1 mutant.

MinE consists of three domains, conferring the functions of membrane binding, MinD binding and the stimulation of ATPase activity, and dimerization ([46], [56], [35], [27], [52], [73]). Membrane binding and dimerization play critical roles in MinE's function, but the functional roles of membrane binding and dimerization are somewhat unclear. During pole-to-pole oscillations *in vivo*, the E ring, a concentrated band of MinE, follows a more diffuse band of MinD from near midcell to cell pole, capping the MinD polar zone ([59], [23], [67]). A disruption in MinE of membrane binding [27] or dimerization [61] inhibits E ring formation, allowing the extension of MinD polar zones. Similarly, a disruption in MinE of membrane binding [44] or dimerization [73] inhibits dynamic pattern formation on a supported lipid bilayer *in vitro*. It has been thought that membrane binding allows a transition between MinD binding events, permitting a MinE dimer to stimulate ATPase activity in multiple MinD dimers before dissociating from the membrane ([52], [53]). The role of dimerization in MinE function is not understood. My results suggest that the functional roles of membrane binding and dimerization are tightly coupled in a stability-switching mechanism, with dimerization underlying stability and membrane binding underlying catastrophe.

4.4 Details of Optimization Using Overlapping-Niche Descent

Here, I describe structural components of overlapping-niche descent for fits to the near-homogeneous data. I describe details pertaining to the implementation of overlapping-niche descent in Appendix G.

4.4.1 Statistical Model

For near-homogeneous MinD and MinE densities at time t_k, $\mathrm{y}_{1,k}$ and $\mathrm{y}_{2,k}$, and corresponding observable model values, $y_{1,k}$ and $y_{2,k}$,

$$\mathrm{y}_{j,k} = y_{j,k} + \varepsilon_{j,k}, \tag{4.10}$$

with errors $\varepsilon_{j,k}$, for $j \in \{1,2\}$ and $k \in \{1,2\ldots,n_t\}$. Errors, $\varepsilon_{j,k}$, consist of modeling errors and data errors. As is visible in Figure F.19 of Section F.4.3, errors in near-homogeneous data are small compared to the ranges of near-homogeneous data; errors range between roughly 5 μm^{-2} and 30 μm^{-2}, as is visible in Figure F.21, and data ranges on the scale of $5 \cdot 10^3$ μm^{-2}. I expect modeling errors to significantly exceed the relatively small errors in near-homogeneous

data. Thus, I expect errors, $\varepsilon_{j,k}$, to consist primarily of modeling errors. Modeling errors are inherently not independent or identically distributed, but without a better a priori distribution, I assume that $\varepsilon_{j,k}$ for $k \in \{1, 2 \ldots, n_t\}$ are independent and identically distributed from a normal distribution with a mean of 0, for each $j \in \{1, 2\}$. The range of near-homogeneous MinD densities is larger than the range of near-homogeneous MinE densities. Thus, to remove bias in fitting from differences in scale, I assume that errors, $\varepsilon_{j,k}$, are proportional to the range of $\mathrm{y}_{j,k}$ for $k \in \{1, 2 \ldots, n_t\}$, $\bar{\mathrm{y}}_j$, for $j \in \{1, 2\}$. Therefore, collectively, I assume that

$$\mathrm{y}_{j,k} = y_{j,k} + \bar{\mathrm{y}}_j \bar{\varepsilon}_{j,k}, \tag{4.11}$$

where $\bar{\varepsilon}_{j,k}$ for $j \in \{1, 2\}$ and $k \in \{1, 2 \ldots, n_t\}$ are independent and identically distributed from a normal distribution with a mean of 0 and a variance of $\bar{\sigma}^2$, $N(0, \bar{\sigma}^2)$. Thus, the likelihood of $y_{j,k}$ given $\mathrm{y}_{j,k}$ and $\bar{\sigma}^2$, for $j \in \{1, 2\}$ and $k \in \{1, 2 \ldots, n_t\}$, is

$$\mathcal{L}(y_{j,k} | \mathrm{y}_{j,k}, \bar{\sigma}^2 : j \in \{1, 2\}, k \in \{1, 2 \ldots, n_t\}) =$$
$$\prod_{j=1}^{2} \prod_{k=1}^{n_t} \frac{1}{\sqrt{2\pi \bar{\mathrm{y}}_j^2 \bar{\sigma}^2}} \exp\left(-\frac{(\mathrm{y}_{j,k} - y_{j,k})^2}{\bar{\mathrm{y}}_j^2 \bar{\sigma}^2} \right) = \bar{C} \exp\left(\frac{1}{\bar{\sigma}^2} \sum_{j=1}^{2} \sum_{k=1}^{n_t} -\frac{(\mathrm{y}_{j,k} - y_{j,k})^2}{\bar{\mathrm{y}}_j^2} \right), \tag{4.12}$$

for constant $\bar{C} > 0$. The values of $y_{j,k}$ for $j \in \{1, 2\}$ and $k \in \{1, 2 \ldots, n_t\}$ that maximize $\mathcal{L}(y_{j,k} | \mathrm{y}_{j,k}, \bar{\sigma}^2 : j \in \{1, 2\}, k \in \{1, 2 \ldots, n_t\})$ are those that minimize

$$\sum_{j=1}^{2} \sum_{k=1}^{n_t} \frac{(\mathrm{y}_{j,k} - y_{j,k})^2}{\bar{\mathrm{y}}_j^2}. \tag{4.13}$$

Thus, I measure the difference in observable model values from the near homogenous data by the sum of weighted squared residuals in equation (4.13).

4.4.2 Defining $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$, $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x})$, and $r_{\Delta x}(\mathbf{p}, \mathbf{x})$

Preliminarily, for consistency with previous notation, I define: $x_1 = c_d$, $x_2 = c_{de}$, and $x_3 = c_e$ for the Modified Bonny Model and the Extended Bonny Model, $x_1 = c_d$, $x_2 = c_{de}$, $x_3 = c_{ede}$, and $x_4 = c_e$ for the Symmetric Activation Model, and $x_1 = c_d$, $x_2 = c_{de}$, $x_3 = c_{ded}$, and $x_4 = c_e$ for the Asymmetric Activation Model; $y_1 =$ the concentration of MinD monomers (μm^{-2}) and $y_2 =$ the concentration of MinE monomers (μm^{-2}); for constant bulk and persistent lipid bilayer-bound MinD and MinE densities, C_d and C_e, as described in Section F.4.4, $g_1 = 2(x_1 + x_2) + C_d$ and $g_2 = 2(x_2 + x_3) + C_e$ for the Modified Bonny Model and the Extended Bonny Model, $g_1 = 2(x_1 + x_2 + x_3) + C_d$ and $g_2 = 2(x_2 + 2x_3 + x_4) + C_e$ for the Symmetric Activation Model, and $g_1 = 2(x_1 + x_2 + 2x_3) + C_d$ and $g_2 = 2(x_2 + x_3 + x_4) + C_e$ for the Asymmetric Activation Model. Additionally, for each model, I uniquely identify each parameter with $p_1, p_2, \ldots, p_{n_p}$.

As described in Section 4.4.1, I measure the difference in observable model values from the

near homogenous data by the sum of weighted squared residuals in equation (4.13). Thus, I define $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ such that

$$r_{\mathrm{y}}(\mathbf{p}, \mathbf{x}) = \frac{1}{\sum_{j=1}^{n_y} \sum_{k=1}^{n_t} \bar{\mathrm{y}}_j^{-2} \mathrm{y}_{j,k}^2} \sum_{j=1}^{n_y} \sum_{k=1}^{n_t} \bar{\mathrm{y}}_j^{-2} \big(\mathrm{y}_{j,k} - g_j(\mathbf{p}, x_{1,k}, \ldots, x_{n_x,k})\big)^2, \tag{4.14}$$

where I normalize by $\sum_{j=1}^{n_y} \sum_{k=1}^{n_t} \bar{\mathrm{y}}_j^{-2} \mathrm{y}_{j,k}^2$ to match the scale of $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ in equation (2.6). I use the data grid, with a data point every 3 s, as the numerical discretization grid. Thus, I define $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x}) = 0$. I define $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ as in equation (2.13a), and discretize models using a Simpson's method finite difference, a finite difference with fourth order accuracy. Thus, in $r_{\Delta x}(\mathbf{p}, \mathbf{x})$,

$$\Delta x_{i,k} = \begin{cases} 0 & \text{if } k \in \{1, n_t\} \\ \dfrac{x_{i,k_+} - x_{i,k_-}}{2\Delta t} & \text{if } k \in \mathcal{I}_\Delta \setminus \{1, n_t\}, \end{cases}$$

$$F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) = \begin{cases} 0 & \text{if } k \in \{1, n_t\} \\ \displaystyle\sum_{m=-1}^{1} b_m F_i\left(\mathbf{p}, x_{1,k+m}, x_{2,k+m}, \ldots, x_{n_x,k+m}\right) & \text{if } k \in \mathcal{I}_\Delta \setminus \{1, n_t\}, \end{cases} \tag{4.15}$$

where k_+ is the index above k in $\mathcal{I}_\Delta$, k_- is the index below k in $\mathcal{I}_\Delta$, $\Delta t = 3$ s is the grid spacing in $\{t_k : k \in \mathcal{I}_\Delta\}$, $b_{-1} = 1/6$, $b_0 = 4/6$, $b_1 = 1/6$, and F_i is as defined in equation (2.7). In smoothing penalties, $s_i(\mathbf{x})$, of $r_{\Delta x}(\mathbf{p}, \mathbf{x})$, I set $\alpha_i = 1$, $\beta_i = 10^2$, and $\gamma_i = 2$, for all $i \in \{1, 2, \ldots, n_x\}$, to insignificantly modify $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ with a smooth set of state values and to strongly penalize $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ with a jagged set of state values.

4.4.3 Domain Restrictions on Parameters and States

As described in Section F.4.4, I restrict nonnegative constant bulk and persistent lipid bilayer-bound MinD and MinE densities, C_d and C_e, such that

$$0 \leq C_d \leq 317.88 \ \mu\mathrm{m}^{-2}, \tag{4.16a}$$

$$0 \leq C_e \leq 249.25 \ \mu\mathrm{m}^{-2}. \tag{4.16b}$$

The nonnegative concentration of persistent lipid bilayer-bound MinD dimers is necessarily no larger than half the constant bulk and persistent lipid bilayer-bound MinD density. Thus, I restrict $c_{\bar{d}}$ such that

$$0 \leq c_{\bar{d}} \leq 158.94 \ \mu\mathrm{m}^{-2}. \tag{4.17}$$

The parameter $c_{\max}$ dictates the maximum concentration of membrane-bound MinD dimers. Thus, I restrict $c_{\max}$ to values greater than or equal to half the maximal near homogeneous MinD density value, $D_{\max}/2$, where $D_{\max} = 1.02 \cdot 10^4 \ \mu m^{-2}$. Additionally, I assume $c_{\max}$ is on

the scale of $D_{\max}/2$, so I bound $c_{\max}$ above by $100 \cdot D_{\max}/2$. Thus, I restrict $c_{\max}$ such that

$$D_{\max}/2 \leq c_{\max} \leq 100 \cdot D_{\max}/2 \tag{4.18}$$

As per condition (4.5), I assume that $n_s \leq D_{\max}/2$. Also, equation (4.3) can be undefined if $n_s = 0$, so I bound n_s below by 1. Thus, I restrict n_s such that

$$1 \leq c_s \leq D_{\max}/2 \tag{4.19}$$

Rate parameters, $\omega^z_{u,v\to x,y}$ for $u, v, x, y, z \in \{\emptyset, D, E, d, de, ede, ded, e\}$, are only biologically relevant if nonnegative. Also, to restrict optimization within ranges of reactions that are not overly fast, I restrict all rate parameters to no more than 10 units. Thus, I restrict rate parameters such that:

$$0 \leq p \leq 10 \ u_p \text{ for all } p \in \left\{\omega^z_{u,v\to x,y} : u, v, x, y, z \in \{\emptyset, D, E, d, de, ede, ded, e\}\right\}, \tag{4.20}$$

where u_p is the units of parameter p. In experiments similar to those of Ivanov and Mizuuchi, in the absence of MinE, MinD dimers dissociate from a supported lipid bilayer at a rate of 0.210 s^{-1} at low concentrations of supported lipid bilayer-bound MinD dimers [73]. Thus, for nonzero $\omega_{d\to D}$, I restrict $\omega_{d\to D}$ to within 50% of 0.210 s^{-1}:

$$0.105 \leq \omega_{d\to D} \leq 0.315 \ s^{-1} \text{ if } \omega_{d\to D} \neq 0. \tag{4.21}$$

Additionally, in experiments similar to those of Ivanov and Mizuuchi, MinE-facilitated dissociation rates of MinD dimers from a supported lipid bilayer were found to be 0.14 s^{-1}, 0.17 s^{-1}, and 0.18 s^{-1} with respective bulk MinE concentrations of 2.5 μM, 3 μM, and 4 μM [73]. Thus, I restrict MinE-facilitated dissociation rates of MinD dimers from the supported lipid bilayer to no less than 50% of 0.14 s^{-1} and no more than 150% of 0.18 s^{-1}:

$$0.07 \leq \omega_{de\to D,E} + \omega_{de\to D,e} \leq 0.27 \ s^{-1}, \tag{4.22a}$$

for the Modified Bonny Model, the Extended Bonny Model, and the Asymmetric Activation Model;

$$0.07 \leq \omega_{ede\to E,D,E} + \omega_{ede\to E,D,e} + \omega_{ede\to D,e,e} \leq 0.27 \ s^{-1}, \tag{4.22b}$$

for the Symmetric Activation Model;

Concentrations c_d, c_{de}, c_{ede}, c_{ded}, and c_e are only biologically relevant if nonnegative. Thus, I restrict c_d, c_{de}, c_{ede}, c_{ded}, and c_e to nonnegative values:

$$c_{i,k} \geq 0 \text{ for all } i \in \{d, de, ede, ded, e\} \text{ and } k \in \mathcal{I}_\Delta, \tag{4.23}$$

where $c_{d,k}$, $c_{de,k}$, $c_{ede,k}$, $c_{ded,k}$, and $c_{e,k}$ are the values of c_d, c_{de}, c_{ede}, c_{ded}, and c_e at the k^{th} index of the numerical discretization. Details of overlapping-niche descent on restricted domains are described in Section C.2.3.

4.4.4 Niches

I choose values of λ to define niches as in Section 3.4.3. For convenience, I repeat the discussion from Section 3.4.3 below. I choose 101 values of λ, λ_k for $k = 1, 2, \ldots, 101$, to define 101 niches. The bounds (B.68) and (B.69), which state that $\breve{r}_{\text{y}}(\lambda) \leq \bar{\varepsilon}$ if $\lambda \leq \bar{\varepsilon}/(1+\bar{\varepsilon})$ and $\breve{r}_{\Delta x}(\lambda) \leq \bar{\varepsilon}$ if $\lambda \geq 1/(1+\bar{\varepsilon})$ for some tolerance $\bar{\varepsilon}$, provide a meaningful guide for the choice of λ_k. Thus, based on the bounds (B.68) and (B.69) with chosen $\bar{\varepsilon} = b^0, b^{-1}, \ldots, b^{-50}$ and base b such that $b^{-50} = 10^{-6}$, I define λ_k for $k = 1, 2, \ldots, 101$ such that

$$\lambda_k = \begin{cases} \dfrac{b^{51-k}}{1+b^{51-k}} & \text{if } k \leq 51 \\ \dfrac{1}{1+b^{51-k}} & \text{if } k > 51. \end{cases} \tag{4.24}$$

My choice of λ_k distributes the values of λ_k for $k = 1, 2, \ldots, 101$ more densely near 0 and 1 and less densely near 0.5. For reference, $\lambda_1 \approx 10^{-6}$, $\lambda_2 \approx 1.3 \cdot 10^{-6}$, $\lambda_{51} = 0.5$, $\lambda_{52} \approx 0.57$, $\lambda_{100} \approx 1 - 1.3 \cdot 10^{-6}$, and $\lambda_{101} \approx 1 - 10^{-6}$.

4.4.5 Calculating Confidence Intervals

I calculate confidence intervals as in Section 3.4.4. For convenience, I repeat the discussion from Section 3.4.4 below. I calculate confidence intervals by bootstrapping, given the complex nonlinear relationship between data noise and parameter noise that would not be adequately captured using a (Taylor expansion based) delta method [39]. In doing so, I calculate observable-state residuals,

$$\tilde{\varepsilon}_{j,k} = \text{y}_{j,k} - g_j(\tilde{\mathbf{p}}, \tilde{x}_{1,k}, \ldots, \tilde{x}_{n_x,k}), \tag{4.25}$$

where $\tilde{\mathbf{p}} = \tilde{\mathbf{p}}^{\lambda_{101}}$ and $\tilde{\mathbf{x}} = \tilde{\mathbf{x}}^{\lambda_{101}}$, the parameter and state values that minimize $r(\mathbf{p}, \mathbf{x}; \lambda_{101})$, and $\tilde{x}_{i,k}$ is the value in $\tilde{\mathbf{x}}$ from the i^{th} state and the k^{th} grid index, for $i \in \{1, 2, \ldots, n_x\}$, $j \in \{1, 2, \ldots, n_y\}$, and $k \in \{1, 2, \ldots, n_t\}$. By resampling residuals, I generate $n_b = 10^3$ bootstrap data sets:

$$\text{y}_{j,k} = g_j(\tilde{\mathbf{p}}, \tilde{x}_{1,k}, \ldots, \tilde{x}_{n_x,k}) + \tilde{\varepsilon}_{j,l}, \tag{4.26}$$

where l is randomly sampled with replacement from $\{1, 2, \ldots, n_t\}$, for $j \in \{1, 2, \ldots, n_y\}$ and $k \in \mathcal{I}_\Delta$. I replace observed data values in $r(\mathbf{p}, \mathbf{x}; \lambda)$ with bootstrap data values from the i^{th} bootstrap data set to construct the functional $r_i^b(\mathbf{p}, \mathbf{x}; \lambda)$. Globally minimizing $r_i^b(\mathbf{p}, \mathbf{x}; \lambda)$ using overlapping-niche descent for all $i \in \{1, 2, \ldots, n_b\}$ would be computationally prohibitive. Rather,

if residuals are not overly large, the optimal parameters and state values of $r_i^b(\mathbf{p}, \mathbf{x}; \lambda)$ will generally be fairly similar to $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{x}}$. Thus, with $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{x}}$ as initial parameters and state values, I locally optimize $r_i^b(\mathbf{p}, \mathbf{x}; \lambda_b)$ using accelerated descent, for all $i \in \{1, 2, \ldots, n_b\}$, with λ_b chosen large enough to weight local optimization towards a numerical solution but not so large that $\mathbf{p}$ and $\mathbf{x}$ are fixed near $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{x}}$. Specifically, I choose

$$\lambda_b = \arg\min \left\{ \left| r_{\mathrm{y}}(\tilde{\mathbf{p}}^\lambda, \tilde{\mathbf{x}}^\lambda) - 10^3 r_{\Delta x}(\tilde{\mathbf{p}}^\lambda, \tilde{\mathbf{x}}^\lambda) \right| : \lambda \in \{\lambda_1, \lambda_2, \ldots, \lambda_{101}\} \right\}. \tag{4.27}$$

From the n_b local optimizations, I construct a distribution of values for each parameter. From the distribution of values for parameter p_j, I compute the 2.5^{th} and 97.5^{th} percentile values, which I translate into the 95% confidence interval for parameter p_j, for $j \in \{1, 2, \ldots, n_p\}$.

4.5 Discussion

In this chapter, I briefly summarized extracting time-course data for model fitting from experimental measurements of the Min system. Then, I fit established and novel biochemistry-based models to the time-course data using my parameter estimation method for differential equations. Comparing models to the time-course data allowed me to make precise distinctions between biochemical assumptions in the various models. My modeling and fitting supported a novel model that accounts for MinE's previously unmodeled dual role as a stabilizer and an inhibitor of MinD membrane binding. It suggests specific biological functions for MinE membrane binding and dimerization, which play critical but somewhat unclear roles in Min system dynamics.

In my supported model, MinD and MinE form an unstable complex on the membrane, where one MinE dimer is bound to one MinD dimer, and a stable complex on the membrane, where one MinE dimer bridges two MinD dimers. Acting as an instability switch, MinE dimers that are unbound to MinD on the membrane bind to stable MinD-MinE complexes to form unstable MinD-MinE complexes. Pushing the system towards stability, MinD dimers that are unbound to MinE on the membrane bind to unstable MinD-MinE complexes to form stable MinD-MinE complexes and bind to membrane-bound MinE dimers to inhibit the instability switch. As such, the concentration of MinD dimers that are unbound to MinE on the membrane modulates local stability or catastrophe in the aggregation of MinD and MinE on the membrane. MinE only associates with the membrane by binding to membrane-bound MinD, and MinE binding to membrane-bound MinD decreases the concentration of MinD dimers that are unbound to MinE on the membrane, which, when concentrations are sufficiently low, signals a switch from local stability to catastrophe in the aggregation of MinD and MinE on the membrane. Ultimately, my supported model suggests a regular, ordered, stability-switching mechanism that may underlie the emergent, dynamic behavior of the Min system.

To be reliable, a biological signal should be regular and ordered; to be informative, a biological signal should be sensitive to variability in its stimuli. My proposed stability-switching mechanism is regular and ordered, with a switch between stability and catastrophe, which

would provide sensitivity to local variation in Min system dynamics. As such, my proposed stability-switching mechanism could provide a reliable signal that fluidly transduces local Min system dynamics into a global cellular signal.

Chapter 5

Conclusion

5.1 Summary of Results

- In Chapter 2, I developed a method that allowed me to calculate the optimal data-fitting numerical solution and its parameters for a differential equation model without using numerical integration. Additionally, I showed that my method admits conservation principles and integral representations that allow me to gauge the accuracy of my optimization.

- In Chapter 3, I tested my method on synthetic data and a system of first order ordinary differential equations, a system of second order ordinary differential equations, and a system of partial differential equations. I found that my method accurately identified the optimal data-fitting numerical solution and its parameters in all three contexts. I compared the performance of my method to that of an analogous numerical-integration-based method, and found that my method identified the optimal data-fitting numerical solution more robustly than the analogous numerical-integration-based method, while requiring significantly less time to do so. I also explored an example where my method informed modeling insufficiencies and potential model improvements for an incomplete variant of a model. Finally, I showed that my optimization routine converged to values that were consistent with my derived conservation principles and integral representations.

- In Chapter 4, I briefly summarized extracting time-course data for model fitting from experimental measurements of the Min system and fit established and novel biochemistry-based models to the time-course data using my method. Comparing the models to time-course data allowed me to make precise distinctions between biochemical assumptions in the various models. My modeling and fitting supported a novel model that accounts for MinE's previously unmodeled dual role as a stabilizer and an inhibitor of MinD membrane binding. It suggests that a regular, ordered, stability-switching mechanism underlies the emergent, dynamic behavior of the Min system.

5.2 Limitations of Overlapping-Niche Descent

I designed my method for global optimization with complex systems of differential equations. When computing descent in parallel on a computer cluster, my method has shown itself to be fairly computationally efficient. However, my method, as specified, could be computationally

prohibitive when computing descent in serial on a single computer. When testing my method, I found that for a relatively complex system of differential equations, the Bonny model with spatially homogeneous conditions, as defined in Equation 3.2, overlapping-niche descent would converge to the optimal data-fitting numerical solution in several generations using only a few niches. However, with only a few niches, conservation principles and integral representations are inaccurately calculated using numerical integration, and thus are not informative in gauging the accuracy and progress of optimization. Alternatively, for more robust optimization when computing descent in serial on a single computer, I suggest applying overlapping-niche descent with many niches but only applying descent to several chosen individuals in each generation. This approach will be slower than applying descent to all individuals in each generation when computing descent in parallel on a computer cluster, but selection across niches will still contribute to the synergistic minimization amongst individuals in different niches, and conservation principles and integral representations will be accurately calculated.

5.3 Extensions of the Homotopy-Minimization Method

In Section 3.7, I demonstrated how parameters and state values from overlapping-niche descent as $\lambda \to 0^+$ could inform model shortcomings and potential model improvements. My demonstration was a proof of concept that was qualitative in nature, requiring model improvements to be inferred simply by looking at a graph. The premise of my demonstration could be extended to develop a systematic method for informing model shortcomings and potential model improvements. Such a method would be a valuable tool for refining models.

My homotopy-minimization method, with conservation principles, integral representations, and overlapping-niche descent, applies to any differentiable function of the form $h(\mathbf{v};\lambda) = (1-\lambda)h_1(\mathbf{v}) + \lambda h_2(\mathbf{v})$, where $\min h_1(\mathbf{v}) = 0$ and $\min h_2(\mathbf{v}) = 0$, with variable vector $\mathbf{v}$. Thus, the method naturally extends to a wide class of optimization problems. For example, where $h_1(\mathbf{v})$ is a measure of how well state values fit data and $h_2(\mathbf{v})$ is a measure of how well state values satisfy a system of difference equations, my method naturally extends to find the parameter values of the solution to the system of difference equations that fits data best. More generally, my homotopy-minimization method naturally extends to any constrained optimization problem that minimizes $h_1(\mathbf{v})$ subject to constraints that can be formulated in a functional, $h_2(\mathbf{v})$.

5.4 Limitations in Fitting Models to Spatially Near-Homogeneous Min Data

In Chapter 4, I fit models to spatially near-homogeneous Min data to compare how well models could describe the data. My fitting suggested that the Asymmetric Activation Model, as discussed in Section 4.3.3, could describe the near-homogeneous data best. In Section 4.3.3, I explored which reactions in the Asymmetric Activation Model were indispensable for describing

the near-homogeneous data and found that quite a few reactions were superfluous. In doing so, I found that the Asymmetric Activation Model would admit numerical solutions that fit data almost identically well for a fairly large range of parameter values (not shown). As such, my asymmetric activation model requires refinements in included reactions and parameter estimates for biological realism. Fitting the Asymmetric Activation Model to the near-homogeneous data with pairwise null reactions would allow me to determine how well pairwise reductions of the Asymmetric Activation Model could describe the near-homogeneous data. Such an analysis would allow me to determine the most reduced form of the Asymmetric Activation Model that describes the near-homogeneous data well. As discussed in Section 4.4.3, I was able to confine some parameters to biologically realistic values by using parameter restrictions from experimental measurements. New experimental measurements, such as the dissociation rate of MinD at various concentrations from the supported lipid bilayer in the absence of MinE, will hopefully provide new parameter restrictions, which will allow me to confine parameter estimates to more biologically realistic values in future fitting of the Asymmetric Activation Model to the near-homogeneous data.

5.5 Extensions of Fitting Models to Spatially Near-Homogeneous Min Data

To unravel the local reaction mechanism of the Min system, I focused on fitting ordinary differential equation models to spatially near-homogeneous Min data. My modeling and fitting supported the Asymmetric Activation Model, which suggests that a regular, ordered, stability-switching mechanism underlies the emergent, dynamic behavior of the Min system. However, it is still unclear how local asymmetric activation reactions collectively contribute to dynamic pattern formation in Min protein bands on spatial scales that are much larger than the size of an individual Min protein. To address this, I would extend the Asymmetric Activation Model from a system of ordinary differential equations to a system of partial differential equations that describe how Min protein concentrations evolve in space and time. Then, using my homotopy-minimization method, I would fit the extended model to experimental measurements of traveling-wave Min protein bands on a supported lipid bilayer. Such modeling and fitting could aid in unraveling how the Min system transduces local interactions into a global signal.

My homotopy-minimization method could provide a novel computational assay for site-directed mutagenesis experiments, allowing me to map dynamic function in proteins to specific amino acids. In this homotopy-minimization mapping method, I would alter a single protein residue by site directed mutagenesis. Then, using reproducible experiments, like those of Ivanov and Mizuuchi, I would measure dynamic protein behavior. Ultimately, I would fit a model to the measurements, like fitting the Asymmetric Activation Model to the near-homogeneous data, and would map changes in parameters from those fitted with the wild type protein to the amino acid that I mutated. Other site-directed mutagenesis assays measure disruption of a specific

protein function and often require deleterious mutations, which dramatically affect protein function. This homotopy-minimization mapping method would simultaneously map effects from a mutation across multiple functional states of a protein and would require mutations that only mildly alter protein function. It could also provide insight into dynamic protein function that is otherwise difficult to assay, such as cooperative binding. Ultimately, this homotopy-minimization mapping method could provide a useful bridge between modeling and experiments that could contribute to our understanding of the dynamic structure of proteins.

Bibliography

[1] Satya Nanda Vel Arjunan and Masaru Tomita. A new multicompartmental reaction-diffusion modeling method links transient membrane attachment of E. coli MinE to E-ring formation. *Systems and synthetic biology*, 4(1):35–53, 2010.

[2] Daniel Axelrod, Thomas P Burghardt, and Nancy L Thompson. Total internal reflection fluorescence. *Annual review of biophysics and bioengineering*, 13(1):247–268, 1984.

[3] Erfei Bi and J Lutkenhaus. Cell division inhibitors SulA and MinCD prevent formation of the FtsZ ring. *Journal of bacteriology*, 175(4):1118–1125, 1993.

[4] Mike Bonny, Elisabeth Fischer-Friedrich, Martin Loose, Petra Schwille, and Karsten Kruse. Membrane binding of mine allows for a comprehensive description of min-protein pattern formation. *PLoS Comput Biol*, 9(12):e1003347, 2013.

[5] Peter Borowski and Eric N Cytrynbaum. Predictions from a stochastic polymer model for the MinDE protein dynamics in Escherichia coli. *Physical Review E*, 80(4):041916, 2009.

[6] James P. Boyle and Richard L. Dykstra. A method for finding projections onto the intersection of convex sets in hilbert spaces. In Richard Dykstra, Tim Robertson, and Farroll T. Wright, editors, *Advances in Order Restricted Statistical Inference*, pages 28–47, New York, NY, 1986. Springer New York.

[7] D.A. Campbell and O. Chkrebtii. Maximum profile likelihood estimation of differential equation parameters through model based smoothing state estimates. *Mathematical Biosciences*, 246(2):283 – 292, 2013.

[8] David Campbell and Russell J. Steele. Smooth functional tempering for nonlinear differential equation models. *Statistics and Computing*, 22(2):429–443, 2012.

[9] David A. Campbell, Giles Hooker, and Kim B. McAuley. Parameter estimation in differential equation models with constrained states. *Journal of Chemometrics*, 26(6):322–332, 2012.

[10] Jiguo Cao and James O. Ramsay. Parameter cascades and profiling in functional data analysis. *Computational Statistics*, 22(3):335–351, 2007.

[11] J.P. Chandler, Doyle E. Hill, and H.Olin Spivey. A program for efficient integration of rate equations and least-squares fitting of chemical reaction data. *Computers and Biomedical Research*, 5(5):515 – 534, 1972.

[12] Brian D Corbin, XuanChuan Yu, and William Margolin. Exploring intracellular space: function of the Min system in roundshaped Escherichia coli. *The EMBO journal*, 21(8):1998–2008, 2002.

[13] Eric N Cytrynbaum and Brandon DL Marshall. A multistranded polymer model explains MinDE dynamics in E. coli cell division. *Biophysical journal*, 93(4):1134–1150, 2007.

[14] PA De Boer, Robin E Crossley, and Lawrence I Rothfield. Roles of MinC and MinD in the site-specific septation block mediated by the MinCDE system of Escherichia coli. *Journal of Bacteriology*, 174(1):63–70, 1992.

[15] Piet A.J. de Boer, Robin E. Crossley, and Lawrence I. Rothfield. A division inhibitor and a topological specificity factor coded for by the minicell locus determine proper placement of the division septum in e. coli. *Cell*, 56(4):641 – 649, 1989.

[16] Frank Deutsch and Hein Hundal. The rate of convergence of dykstra's cyclic projections algorithm: The polyhedral case. *Numerical Functional Analysis and Optimization*, 15(5-6):537–565, 1994.

[17] Barbara Di Ventura and Victor Sourjik. Selforganized partitioning of dynamically localized proteins in bacterial cell division. *Molecular systems biology*, 7(1):457, 2011.

[18] Daniel M. Dunlavy and Dianne P. O'Leary. *Homotopy optimization methods for global optimization.* United States. Dept. of Energy, 2005.

[19] Richard L. Dykstra. An algorithm for restricted least squares regression. *Journal of the American Statistical Association*, 78(384):837–842, 1983.

[20] David Fange and Johan Elf. Noise-induced Min phenotypes in E. coli. *PLoS Comput Biol*, 2(6):e80, 2006.

[21] Elisabeth Fischer-Friedrich, Giovanni Meacci, Joe Lutkenhaus, Hugues Chaté, and Karsten Kruse. Intra-and intercellular fluctuations in Min-protein dynamics decrease with cell length. *Proceedings of the National Academy of Sciences*, 107(14):6134–6139, 2010.

[22] Jacob Halatek and Erwin Frey. Highly canalized MinD transfer and MinE sequestration explain the origin of robust MinCDE-protein dynamics. *Cell Reports*, 1(6):741–752, 2012.

[23] Cynthia A Hale, Hans Meinhardt, and Piet AJ de Boer. Dynamic localization cycle of the cell division regulator MinE in Escherichia coli. *The EMBO journal*, 20(7):1563–1572, 2001.

[24] Max Hoffmann and Ulrich S Schwarz. Oscillations of Min-proteins in micropatterned environments: a three-dimensional particle-based stochastic simulation approach. *Soft Matter*, 10(14):2388–2396, 2014.

[25] Martin Howard and Andrew D Rutenberg. Pattern formation inside bacteria: fluctuations due to the low copy number of proteins. *Physical Review Letters*, 90(12):128102, 2003.

[26] Martin Howard, Andrew D Rutenberg, and Simon de Vet. Dynamic compartmentalization of bacteria: accurate division in E. coli. *Physical review letters*, 87(27):278102, 2001.

[27] ChengWei Hsieh, TiYu Lin, HsinMei Lai, ChuChi Lin, TingSung Hsieh, and YuLing Shih. Direct MinE–membrane interaction contributes to the proper localization of MinDE in E. coli. *Molecular microbiology*, 75(2):499–512, 2010.

[28] Zonglin Hu, Edward P Gogol, and Joe Lutkenhaus. Dynamic assembly of MinD on phospholipid vesicles regulated by ATP and MinE. *Proceedings of the National Academy of Sciences*, 99(10):6761–6766, 2002.

[29] Zonglin Hu and Joe Lutkenhaus. Topological regulation of cell division in Escherichia coli involves rapid pole to pole oscillation of the division inhibitor MinC under the control of MinD and MinE. *Molecular microbiology*, 34(1):82–90, 1999.

[30] Zonglin Hu and Joe Lutkenhaus. Analysis of MinC reveals two independent domains involved in interaction with MinD and FtsZ. *Journal of bacteriology*, 182(14):3965–3971, 2000.

[31] Zonglin Hu and Joe Lutkenhaus. Topological regulation of cell division in E. coli: spatiotemporal oscillation of MinD requires stimulation of its ATPase by MinE and phospholipid. *Molecular cell*, 7(6):1337–1343, 2001.

[32] Zonglin Hu and Joe Lutkenhaus. A conserved sequence at the Cterminus of MinD is required for binding to the membrane and targeting MinC to the septum. *Molecular microbiology*, 47(2):345–355, 2003.

[33] Zonglin Hu, Amit Mukherjee, Sebastien Pichoff, and Joe Lutkenhaus. The MinC component of the division site selection system in Escherichia coli interacts with FtsZ to prevent polymerization. *Proceedings of the National Academy of Sciences*, 96(26):14819–14824, 1999.

[34] Zonglin Hu, Cristian Saez, and Joe Lutkenhaus. Recruitment of MinC, an inhibitor of Z-ring formation, to the membrane in Escherichia coli: role of MinD and MinE. *Journal of bacteriology*, 185(1):196–203, 2003.

[35] Jian Huang, Chune Cao, and Joe Lutkenhaus. Interaction between FtsZ and inhibitors of cell division. *Journal of Bacteriology*, 178(17):5080–5085, 1996.

[36] Kerwyn Casey Huang, Yigal Meir, and Ned S Wingreen. Dynamic structures in Escherichia coli: spontaneous formation of MinE rings and MinD polar zones. *Proceedings of the National Academy of Sciences*, 100(22):12724–12728, 2003.

[37] Kerwyn Casey Huang and Ned S Wingreen. Min-protein oscillations in round bacteria. *Physical biology*, 1(4):229, 2004.

[38] Vassili Ivanov and Kiyoshi Mizuuchi. Multiple modes of interconverting dynamic pattern formation by bacterial cell division proteins. *Proceedings of the National Academy of Sciences*, 107(18):8071–8078, 2010.

[39] M. Joshi, A. Seidel-Morgenstern, and A. Kremling. Exploiting the bootstrap method for quantifying parameter confidence intervals in dynamical systems. *Metabolic Engineering*, 8(5):447 – 455, 2006.

[40] Rex A Kerr, Herbert Levine, Terrence J Sejnowski, and Wouter-Jan Rappel. Division accuracy in a stochastic model of Min oscillations in Escherichia coli. *Proceedings of the National Academy of Sciences of the United States of America*, 103(2):347–352, 2006.

[41] Karsten Kruse. A dynamic model for determining the middle of Escherichia coli. *Biophysical journal*, 82(2):618–627, 2002.

[42] Laura L Lackner, David M Raskin, and Piet AJ de Boer. ATP-dependent interactions between Escherichia coli Min proteins and the phospholipid membrane in vitro. *Journal of bacteriology*, 185(3):735–749, 2003.

[43] Hua Liang and Hulin Wu. Parameter estimation for differential equation models using a framework of measurement error in regression models. *Journal of the American Statistical Association*, 103(484):1570–1583, 2008. PMID: 19956350.

[44] Martin Loose, Elisabeth Fischer-Friedrich, Christoph Herold, Karsten Kruse, and Petra Schwille. Min protein patterns emerge from rapid rebinding and membrane interaction of mine. *Nat Struct Mol Biol*, 18(5):577–583, 05 2011.

[45] Martin Loose, Elisabeth Fischer-Friedrich, Jonas Ries, Karsten Kruse, and Petra Schwille. Spatial regulators for bacterial cell division self-organize into surface waves in vitro. *Science*, 320(5877):789–792, 2008.

[46] Lu-Yan Ma, Glenn King, and Lawrence Rothfield. Mapping the MinE site involved in interaction with the MinD division site selection protein of Escherichia coli. *Journal of Bacteriology*, 185(16):4948–4955, 2003.

[47] Giovanni Meacci and Karsten Kruse. Min-oscillations in Escherichia coli induced by interactions of membrane-bound proteins. *Physical Biology*, 2(2):89, 2005.

[48] Hans Meinhardt and Piet AJ de Boer. Pattern formation in Escherichia coli: a model for the pole-to-pole oscillations of Min proteins and the localization of the division site. *Proceedings of the National Academy of Sciences*, 98(25):14202–14207, 2001.

[49] Hongyu Miao, Carrie Dykes, Lisa M. Demeter, and Hulin Wu. Differential equation modeling of hiv viral fitness experiments: Model identification, model selection, and multimodel inference. *Biometrics*, 65(1):292–300, 2009.

[50] Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/k2). *Doklady an SSSR*, 269(3):543–547, 1983.

[51] Brendan O'Donoghue and Emmanuel Candès. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, 15(3):715–732, 2015.

[52] Kyung-Tae Park, Wei Wu, Kevin P Battaile, Scott Lovell, Todd Holyoak, and Joe Lutkenhaus. The Min oscillator uses MinD-dependent conformational changes in MinE to spatially regulate cytokinesis. *Cell*, 146(3):396–407, 2011.

[53] KyungTae Park, Wei Wu, Scott Lovell, and Joe Lutkenhaus. Mechanism of the asymmetric activation of the MinD ATPase by MinE. *Molecular microbiology*, 85(2):271–281, 2012.

[54] Zdeněk Petrášek and Petra Schwille. Simple membrane-based model of the Min oscillator. *New Journal of Physics*, 17(4):043023, 2015.

[55] Sebastien Pichoff and Joe Lutkenhaus. Escherichia coli division inhibitor MinCD blocks septation by preventing Z-ring formation. *Journal of bacteriology*, 183(22):6630–6635, 2001.

[56] Sébastien Pichoff, Benedikt Vollrath, Christian Touriol, and JeanPierre Bouché. Deletion analysis of gene minE which encodes the topological specificity factor of cell division in Escherichia coli. *Molecular microbiology*, 18(2):321–329, 1995.

[57] A.A. Poyton, M.S. Varziri, K.B. McAuley, P.J. McLellan, and J.O. Ramsay. Parameter estimation in continuous-time dynamic models using principal differential analysis. *Computers and Chemical Engineering*, 30(4):698 – 708, 2006.

[58] J. O. Ramsay, G. Hooker, D. Campbell, and J. Cao. Parameter estimation for differential equations: a generalized smoothing approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(5):741–796, 2007.

[59] David M. Raskin and Piet A.J. de Boer. The MinE ring: An FtsZ-independent cell structure required for selection of the correct division site in E. coli. *Cell*, 91(5):685 – 694, 1997.

[60] David M Raskin and Piet AJ de Boer. Rapid pole-to-pole oscillation of a protein required for directing division to the middle of escherichia coli. *Proceedings of the National Academy of Sciences*, 96(9):4971–4976, 1999.

[61] S. L. Rowland, X. Fu, M. A. Sayed, Y. Zhang, W. R. Cook, and L. I. Rothfield. Membrane redistribution of the Escherichia coli MinD protein induced by MinE. *Journal of Bacteriology*, 182(3):613–619, 2000.

[62] Jeff B Schulte, Rene W Zeto, and David Roundy. Theoretical prediction of disrupted min oscillation in flattened Escherichia coli. *PloS one*, 10(10):e0139813, 2015.

[63] Jakob Schweizer, Martin Loose, Mike Bonny, Karsten Kruse, Ingolf Mönch, and Petra Schwille. Geometry sensing by self-organized protein patterns. *Proceedings of the National Academy of Sciences*, 109(38):15283–15288, 2012.

[64] Supratim Sengupta, Julien Derr, Anirban Sain, and Andrew D Rutenberg. Stuttering Min oscillations within E. coli bacteria: a stochastic polymerization model. *Physical biology*, 9(5):056003, 2012.

[65] Supratim Sengupta and Andrew Rutenberg. Modeling partitioning of Min proteins between daughter cells after septation in Escherichia coli. *Physical biology*, 4(3):145, 2007.

[66] Yu-Ling Shih, Kai-Fa Huang, Hsin-Mei Lai, Jiahn-Haur Liao, Chai-Siah Lee, Chiao-Min Chang, Huey-Ming Mak, Cheng-Wei Hsieh, and Chu-Chi Lin. The N-terminal amphipathic helix of the topological specificity factor MinE is associated with shaping membrane curvature. *PloS one*, 6(6):e21425, 2011.

[67] YuLing Shih, Xiaoli Fu, Glenn F King, Trung Le, and Lawrence Rothfield. Division site placement in E. coli: mutations that prevent formation of the MinE ring lead to loss of the normal midcell arrest of growth of polar MinD membrane domains. *The EMBO journal*, 21(13):3347–3357, 2002.

[68] YuLing Shih, Ikuro Kawagishi, and Lawrence Rothfield. The MreB and Min cytoskeletallike systems play independent roles in prokaryotic polar differentiation. *Molecular microbiology*, 58(4):917–928, 2005.

[69] Oleksii Sliusarenko, Jennifer Heinritz, Thierry Emonet, and Christine JacobsWagner. Highthroughput, subpixel precision analysis of bacterial morphogenesis and intracellular spatiotemporal dynamics. *Molecular microbiology*, 80(3):612–627, 2011.

[70] Filipe Tostevin and Martin Howard. A stochastic model of Min oscillations in Escherichia coli and Min protein segregation during cell division. *Physical biology*, 3(1):1, 2005.

[71] J. M. Varah. A spline least squares method for numerical parameter estimation in differential equations. *SIAM Journal on Scientific and Statistical Computing*, 3(1):28–46, 1982.

[72] Archana Varma, Kerwyn Casey Huang, and Kevin D Young. The Min system as a general cell geometry detection mechanism: branch lengths in Y-shaped Escherichia coli cells affect Min oscillation patterns and division dynamics. *Journal of bacteriology*, 190(6):2106–2117, 2008.

[73] Anthony G Vecchiarelli, Min Li, Michiyo Mizuuchi, Ling Chin Hwang, Yeonee Seol, Keir C Neuman, and Kiyoshi Mizuuchi. Membrane-bound MinDE complex acts as a toggle switch

that drives Min oscillation coupled to cytoplasmic depletion of MinD. *Proceedings of the National Academy of Sciences*, 113(11):E1479–E1488, 2016.

[74] Anthony G Vecchiarelli, Min Li, Michiyo Mizuuchi, and Kiyoshi Mizuuchi. Differential affinities of MinD and MinE to anionic phospholipid influence Min patterning dynamics in vitro. *Molecular microbiology*, 93(3):453–463, 2014.

[75] James C Walsh, Christopher N Angstmann, Iain G Duggin, and Paul MG Curmi. Molecular interactions of the Min protein system reproduce spatiotemporal patterning in growing and dividing Escherichia coli cells. *PloS one*, 10(5):e0128148, 2015.

[76] Wei Wu, KyungTae Park, Todd Holyoak, and Joe Lutkenhaus. Determination of the structure of the MinD–ATP complex reveals the orientation of MinD on the membrane and the relative location of the binding sites for MinE and MinC. *Molecular microbiology*, 79(6):1515–1528, 2011.

[77] Hongyu Zhao Xin Qi. Asymptotic efficiency and finite-sample properties of the generalized profiling estimation of parameters in ordinary differential equations. *The Annals of Statistics*, 38(1):435–481, 2010.

[78] Hongqi Xue, Hongyu Miao, and Hulin Wu. Sieve estimation of constant and time-varying coefficients in nonlinear ordinary differential equation models by considering both numerical error and measurement error. *Annals of statistics*, 38(4):2351–2387, 01 2010.

[79] Xiaolei Xun, Jiguo Cao, Bani Mallick, Arnab Maity, and Raymond J. Carroll. Parameter estimation of partial differential equation models. *Journal of the American Statistical Association*, 108(503):1009–1020, 2013.

[80] XuanChuan Yu and William Margolin. FtsZ ring clusters in min and partition mutants: role of both the Min system and the nucleoid in regulating FtsZ ring localization. *Molecular microbiology*, 32(2):315–326, 1999.

[81] Huaijin Zhou, Ryan Schulze, Sandra Cox, Cristian Saez, Zonglin Hu, and Joe Lutkenhaus. Analysis of MinD mutations reveals residues required for MinE stimulation of the MinD ATPase and residues required for MinC interaction. *Journal of Bacteriology*, 187(2):629–638, 2005.

[82] Katja Zieske and Petra Schwille. Reconstitution of self-organizing protein gradients as spatial cues in cell-free systems. *Elife*, 3:e03949, 2014.

Appendix A

Extensions of the Homotopy-Minimization Method Beyond Systems of First Order Ordinary Differential Equations

Here, I extend the parameter estimation method described for systems of first order ordinary differential equations in Sections 2.2 and 2.6 to systems of higher order ordinary differential equations and systems of partial differential equations.

A.1 Extensions to Systems of Higher Order Ordinary Differential Equations

A higher order ordinary differential equation model of some dynamic process in t, with n_x states, $x_1, x_2, \ldots, x_{n_x}$, n_p parameters, $p_1, p_2, \ldots, p_{n_p}$, and n_y observable states, $y_1, y_2, \ldots, y_{n_y}$, is defined by the system of equations,

$$F_i\left(t, p_1, \ldots, p_{n_p}, x_1, \ldots, x_{n_x}, \frac{dx_1}{dt}, \frac{d^2x_1}{dt^2}, \frac{d^3x_1}{dt^3}, \ldots, \frac{dx_2}{dt}, \frac{d^2x_2}{dt^2}, \ldots\right) = 0, \tag{A.1a}$$

$$y_j = g_j\left(p_1, p_2, \ldots, p_{n_p}, x_1, x_2, \ldots, x_{n_x}\right). \tag{A.1b}$$

For some observed data values, $\mathrm{y}_{1,k}, \mathrm{y}_{2,k}, \ldots, \mathrm{y}_{n_y,k}$, measured at at t_k, for $k \in \{1, 2, \ldots, n_t\}$, I extend the method described for first order ordinary differential equation models in Sections 2.2 and 2.6, to find the parameters $p_1, p_2, \ldots, p_{n_p}$ of the numerical solution to the differential equation model with the observable state values that most closely approximate the observed data, in some sense. As with first order ordinary differential equation models, once a numerical method is chosen, equation (A.1a) can be formulated into a method-dependent system of equations for the discrete numerical solution values $x_{i,k}$:

$$f_{i,k}\left(t_1, \ldots, t_{n_t}, p_1, \ldots, p_{n_p}, x_{1,1}, x_{2,1}, \ldots, x_{n_x,n_t}\right) = f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) = 0, \tag{A.2}$$

for all $i \in \{1, 2, \ldots, n_x\}$ and for all $k \in \mathcal{I}_\Delta$, the index set of the numerical discretization. Using some interpolation method, I generate interpolated data, $\hat{\mathrm{y}}_{j,k}$ for $j \in \{1, 2, \ldots, n_y\}$, at grid points with indices in $\mathcal{I}_{\hat{\mathrm{y}}} = \mathcal{I}_\Delta \setminus \{1, 2, \ldots, n_t\}$ from observed data values with indices in $\{1, 2, \ldots, n_t\}$. As with first order ordinary differential equation models, I define the functionals $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$, $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x})$, and $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ with the properties that (i) $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x}) \geq 0$, $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x}) \geq 0$, and $r_{\Delta x}(\mathbf{p}, \mathbf{x}) \geq 0$; (ii) $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x}) = 0$ if and only if $g_j(\mathbf{p}, x_{1,k}, \ldots, x_{n_x,k}) = \mathrm{y}_{j,k}$ for all $j \in \{1, 2, \ldots, n_y\}$ and for all $k \in \{1, 2, \ldots, n_t\}$, $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x}) = 0$ if and only if $\mathcal{I}_\Delta = \{1, 2, \ldots, n_t\}$ or $g_j(\mathbf{p}, x_{1,k}, \ldots, x_{n_x,k}) = \hat{\mathrm{y}}_{j,k}$ for all $j \in \{1, 2, \ldots, n_y\}$ and for all $k \in \mathcal{I}_{\hat{\mathrm{y}}}$, and $r_{\Delta x}(\mathbf{p}, \mathbf{x}) = 0$ if and only if $f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) = 0$ for all $i \in \{1, 2, \ldots, n_x\}$ and for all $k \in \mathcal{I}_\Delta$; and (iii) $r_{\mathrm{y}}(\mathbf{p}_1, \mathbf{x}_1) < r_{\mathrm{y}}(\mathbf{p}_2, \mathbf{x}_2)$ implies that $(\mathbf{p}_1, \mathbf{x}_1)$ gives a better fit to the data than does $(\mathbf{p}_2, \mathbf{x}_2)$, $r_{\hat{\mathrm{y}}}(\mathbf{p}_1, \mathbf{x}_1) < r_{\hat{\mathrm{y}}}(\mathbf{p}_2, \mathbf{x}_2)$ implies that $(\mathbf{p}_1, \mathbf{x}_1)$ gives a better fit to the interpolated data than does $(\mathbf{p}_2, \mathbf{x}_2)$, and $r_{\Delta x}(\mathbf{p}_1, \mathbf{x}_1) < r_{\Delta x}(\mathbf{p}_2, \mathbf{x}_2)$ implies that $(\mathbf{p}_1, \mathbf{x}_1)$ satisfies the numerical solution method better than $(\mathbf{p}_2, \mathbf{x}_2)$ does. Then, I combine $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$, $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x})$, and $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ into the single functional $r(\mathbf{p}, \mathbf{x}; \lambda) = (1 - \lambda) r_{\mathrm{y}}(\mathbf{p}, \mathbf{x}) + (1 - \lambda)^2 r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x}) + \lambda r_{\Delta x}(\mathbf{p}, \mathbf{x})$. I describe the construction of $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$, $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x})$, and $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ using a normalized least-squares measure in Section A.1.1. As with first order ordinary differential equation models, minimizing $r(\mathbf{p}, \mathbf{x}; \lambda)$ as $\lambda \to 1^-$ is equivalent to minimizing $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ subject to the constraints $f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) = 0$ for all $i \in \{1, 2, \ldots, n_x\}$ and for all $k \in \mathcal{I}_\Delta$, and finds the parameters and state values of the optimal data-fitting numerical solution.

A.1.1 Defining $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$, $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x})$, and $r_{\Delta x}(\mathbf{p}, \mathbf{x})$

As with first order ordinary differential equation models, I consider systems of higher order ordinary differential equations that are linear in first derivatives of x_i,

$$\frac{dx_i}{dt} = \bar{F}_i \left(t, p_1, \ldots, p_{n_p}, x_1, \ldots, x_{n_x}, \frac{d^2x_1}{dt^2}, \frac{d^3x_1}{dt^3}, \ldots, \frac{d^2x_2}{dt^2}, \ldots \right), \tag{A.3}$$

for $i \in \{1, 2, \ldots, n_x\}$. In terms of F_i as defined in equation (A.1a),

$$F_i = \frac{dx_i}{dt} - \bar{F}_i \left(t, p_1, \ldots, p_{n_p}, x_1, \ldots, x_{n_x}, \frac{d^2x_1}{dt^2}, \frac{d^3x_1}{dt^3}, \ldots, \frac{d^2x_2}{dt^2}, \ldots \right), \tag{A.4}$$

for $i \in \{1, 2, \ldots, n_x\}$. Also, as with first order ordinary differential equation models, I consider finite difference numerical methods, of the form

$$\begin{gathered} \Delta^1 x_{i,k} = \\ F_{i,k}\big(\{ \bar{F}_i(t_k, \mathbf{p}, x_{1,k}, \ldots, x_{n_x,k}, \Delta^2 x_{1,k}, \Delta^3 x_{1,k}, \ldots, \Delta^2 x_{2,k}, \ldots) : k \in \mathcal{I}_\Delta \} \big) = \\ F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}), \end{gathered} \tag{A.5}$$

where $x_{i,k}$ are numerical solution values, $\Delta^n x_{i,k}$ is some method-dependent finite difference discretization of $\frac{d^n x_i}{dt^n}$ at t_k, and $F_{i,k}$ are some method-dependent functions of $\bar{F}_i$ at t_k, for all

$i \in \{1, 2, \ldots, n_x\}$ and for all $k \in \mathcal{I}_\Delta$. In terms $f_{i,k}$ as defined in equation (A.2),

$$f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) = \Delta^1 x_{i,k} - F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}). \tag{A.6}$$

Thus, using a normalized least-squares measure, as described for first order ordinary differential equation models in Sections 2.3, 2.4.1, and 2.6.1,

$$r_{\mathrm{y}}(\mathbf{p}, \mathbf{x}) = \frac{1}{n_y} \sum_{j=1}^{n_y} \left(\frac{1}{\sum_{k=1}^{n_t} w_{j,k} \mathrm{y}_{j,k}^2} \sum_{k=1}^{n_t} w_{j,k} \big(\mathrm{y}_{j,k} - g_j(\mathbf{p}, x_{1,k}, \ldots, x_{n_x,k}) \big)^2 \right), \tag{A.7a}$$

$$r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x}) = \frac{1}{n_y} \sum_{j=1}^{n_y} \left(\frac{\hat{\sigma}}{\sum_{k \in \mathcal{I}_{\hat{\mathrm{y}}}} \hat{w}_{j,k} \hat{\mathrm{y}}_{j,k}^2} \sum_{k \in \mathcal{I}_{\hat{\mathrm{y}}}} \hat{w}_{j,k} \big(\hat{\mathrm{y}}_{j,k} - g_j(\mathbf{p}, x_{1,k}, \ldots, x_{n_x,k}) \big)^2 \right), \tag{A.7b}$$

$$r_{\Delta x}(\mathbf{p}, \mathbf{x}) = \frac{1}{n_x} \sum_{i=1}^{n_x} \left(\frac{s_i(\mathbf{x})}{\sum_{k \in \mathcal{I}_\Delta} (\Delta^1 x_{i,k})^2} \sum_{k \in \mathcal{I}_\Delta} \big(\Delta^1 x_{i,k} - F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) \big)^2 \right), \tag{A.7c}$$

$$s_i(\mathbf{x}) = \alpha_i + \beta_i \left(\frac{4 \sum_{k \in \mathcal{I}_\Delta \setminus \{1, n_t\}} \left(x_{i,k_-} - 2 x_{i,k} + x_{i,k_+} \right)^2}{\sum_{k \in \mathcal{I}_\Delta \setminus \{1, n_t\}} \left(x_{i,k_+} - x_{i,k_-} \right)^2} \right)^{\gamma_i}, \tag{A.7d}$$

for some data-dependent weights, $w_{j,k}$, some interpolated data-dependent weights, $\hat{w}_{j,k}$, scaling parameter $\hat{\sigma}$, as discussed in Section 2.6.1, and smoothing-penalty parameters, $\alpha_i > 0$, $\beta_i \geq 0$, and $\gamma_i \geq 0$, as discussed in Section 2.4.1, where k_- and k_+ are the indices below and above k in $\mathcal{I}_\Delta$.

A.2 Extensions to Systems of Partial Differential Equations

For simplicity in notation, I present a partial differential equation model with two independent variables. Although, the method naturally extends to models with any finite number of independent variables.

A partial differential equation model of some dynamic process in u and v, with n_x states, $x_1, x_2, \ldots, x_{n_x}$, n_p parameters, $p_1, p_2, \ldots, p_{n_p}$, and n_y observable states, $y_1, y_2, \ldots, y_{n_y}$, is defined by the system of equations,

$$F_i \left(u, v, p_1, \ldots, p_{n_p}, x_1, \ldots, x_{n_x}, \frac{\partial x_1}{\partial u}, \frac{\partial x_1}{\partial v}, \frac{\partial^2 x_1}{\partial^2 u}, \frac{\partial^2 x_1}{\partial u \partial v}, \ldots, \frac{\partial x_2}{\partial u}, \frac{\partial x_2}{\partial v}, \ldots \right) = 0, \tag{A.8a}$$

$$y_j = g_j \left(p_1, p_2, \ldots, p_{n_p}, x_1, x_2, \ldots, x_{n_x} \right). \tag{A.8b}$$

For some observed data values, $\mathrm{y}_{1,k,l}, \mathrm{y}_{2,k,l}, \ldots, \mathrm{y}_{n_y,k,l}$, measured at (u_k, v_l), for $k \in \{1, 2, \ldots, n_u\}$ and $l \in \{1, 2, \ldots, n_v\}$, I extend the method described for first order ordinary differential equation models in Sections 2.2 and 2.6, to find the parameters $p_1, p_2, \ldots, p_{n_p}$ of the numerical solution to the differential equation model with the observable state values that most closely approximate the observed data, in some sense. As with first order ordinary differential equation models, once

a numerical method is chosen, equation (A.8a) can be formulated into a method-dependent system of equations for the discrete numerical solution values $x_{i,k,l}$:

$$f_{i,k,l}\left(u_1,\ldots,u_{n_u},v_1,\ldots,v_{n_v},p_1,\ldots,p_{n_p},x_{1,1,1},x_{2,1,1},\ldots,x_{n_x,n_u,n_v}\right) = f_{i,k,l}(\mathbf{u},\mathbf{v},\mathbf{p},\mathbf{x}) = 0, \tag{A.9}$$

for all $i \in \{1,2,\ldots,n_x\}$ and for all $(k,l) \in \mathcal{I}_{\Delta_u} \times \mathcal{I}_{\Delta_v}$, the index set of the numerical discretization. Using some interpolation method, I generate interpolated data, $\hat{y}_{j,k,l}$ for $j \in \{1,2,\ldots,n_y\}$, at grid points with indices in $\mathcal{I}_{\hat{y}_u} \times \mathcal{I}_{\hat{y}_v} = \mathcal{I}_{\Delta_u} \setminus \{1,2,\ldots,n_u\} \times \mathcal{I}_{\Delta_v} \setminus \{1,2,\ldots,n_v\}$ from observed data values with indices in $\{1,2,\ldots,n_u\} \times \{1,2,\ldots,n_v\}$. As with first order ordinary differential equation models, I define the functionals $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$, $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x})$, and $r_{\Delta x}(\mathbf{p},\mathbf{x})$ with the properties that (i) $r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) \geq 0$, $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) \geq 0$, and $r_{\Delta x}(\mathbf{p},\mathbf{x}) \geq 0$; (ii) $r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) = 0$ if and only if $g_j(\mathbf{p},x_{1,k,l},\ldots,x_{n_x,k,l}) = \mathrm{y}_{j,k,l}$ for all $j \in \{1,2,\ldots,n_y\}$ and for all $(k,l) \in \{1,2,\ldots,n_u\}\times\{1,2,\ldots,n_v\}$, $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) = 0$ if and only if $\mathcal{I}_{\Delta_u}\times\mathcal{I}_{\Delta_v} = \{1,2,\ldots,n_u\}\times\{1,2,\ldots,n_v\}$ or $g_j(\mathbf{p},x_{1,k,l},\ldots,x_{n_x,k,l}) = \hat{\mathrm{y}}_{j,k,l}$ for all $j \in \{1,2,\ldots,n_y\}$ and for all $(k,l) \in \mathcal{I}_{\hat{y}_u} \times \mathcal{I}_{\hat{y}_v}$, and $r_{\Delta x}(\mathbf{p},\mathbf{x}) = 0$ if and only if $f_{i,k,l}(\mathbf{u},\mathbf{v},\mathbf{p},\mathbf{x}) = 0$ for all $i \in \{1,2,\ldots,n_x\}$ and for all $(k,l) \in \mathcal{I}_{\Delta_u} \times \mathcal{I}_{\Delta_v}$; and (iii) $r_{\mathrm{y}}(\mathbf{p}_1,\mathbf{x}_1) < r_{\mathrm{y}}(\mathbf{p}_2,\mathbf{x}_2)$ implies that $(\mathbf{p}_1,\mathbf{x}_1)$ gives a better fit to the data than does $(\mathbf{p}_2,\mathbf{x}_2)$, $r_{\hat{\mathrm{y}}}(\mathbf{p}_1,\mathbf{x}_1) < r_{\hat{\mathrm{y}}}(\mathbf{p}_2,\mathbf{x}_2)$ implies that $(\mathbf{p}_1,\mathbf{x}_1)$ gives a better fit to the interpolated data than does $(\mathbf{p}_2,\mathbf{x}_2)$, and $r_{\Delta x}(\mathbf{p}_1,\mathbf{x}_1) < r_{\Delta x}(\mathbf{p}_2,\mathbf{x}_2)$ implies that $(\mathbf{p}_1,\mathbf{x}_1)$ satisfies the numerical solution method better than $(\mathbf{p}_2,\mathbf{x}_2)$ does. Then, I combine $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$, $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x})$, and $r_{\Delta x}(\mathbf{p},\mathbf{x})$ into the single functional $r(\mathbf{p},\mathbf{x};\lambda) = (1-\lambda)r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) + (1-\lambda)^2 r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) + \lambda r_{\Delta x}(\mathbf{p},\mathbf{x})$. I describe the construction of $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$, $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x})$, and $r_{\Delta x}(\mathbf{p},\mathbf{x})$ using a normalized least-squares measure in Section A.2.1. As with first order ordinary differential equation models, minimizing $r(\mathbf{p},\mathbf{x};\lambda)$ as $\lambda \to 1^-$ is equivalent to minimizing $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$ subject to the constraints $f_{i,k,l}(\mathbf{u},\mathbf{v},\mathbf{p},\mathbf{x}) = 0$ for all $i \in \{1,2,\ldots,n_x\}$ and for all $(k,l) \in \mathcal{I}_{\Delta_u} \times \mathcal{I}_{\Delta_v}$, and finds the parameters and state values of the optimal data-fitting numerical solution.

A.2.1 Defining $r_{\mathrm{y}}(\mathbf{p},\mathbf{x})$, $r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x})$, and $r_{\Delta x}(\mathbf{p},\mathbf{x})$

As with first order ordinary differential equation models, I consider systems of partial differential equations that are linear in first derivatives of x_i with respect to u,

$$\frac{\partial x_i}{\partial u} = \bar{F}_i\left(u,v,p_1,\ldots,p_{n_p},x_1,\ldots,x_{n_x},\frac{\partial x_1}{\partial v},\frac{\partial^2 x_1}{\partial^2 u},\frac{\partial^2 x_1}{\partial u \partial v},\ldots,\frac{\partial x_2}{\partial v},\ldots\right), \tag{A.10}$$

for $i \in \{1,2,\ldots,n_x\}$. In terms of F_i as defined in equation (A.8a),

$$F_i = \frac{\partial x_i}{\partial u} - \bar{F}_i\left(u,v,p_1,\ldots,p_{n_p},x_1,\ldots,x_{n_x},\frac{\partial x_1}{\partial v},\frac{\partial^2 x_1}{\partial^2 u},\frac{\partial^2 x_1}{\partial u \partial v},\ldots,\frac{\partial x_2}{\partial v},\ldots\right), \tag{A.11}$$

for $i \in \{1, 2, \ldots, n_x\}$. Also, as with first order ordinary differential equation models, I consider finite difference numerical methods, of the form

$$\begin{aligned}
&\Delta^{1,0} x_{i,k,l} = \\
&F_{i,k,l}\big(\big\{ \bar{F}_i(u_k, v_l, \mathbf{p}, x_{1,k,l}, \ldots, x_{n_x,k,l}, \Delta^{0,1} x_{1,k,l}, \Delta^{2,0} x_{1,k,l}, \ldots) : k, l \in \mathcal{I}_{\Delta_u}, \mathcal{I}_{\Delta_v} \big\} \big) \\
&\qquad = F_{i,k,l}(\mathbf{u}, \mathbf{v}, \mathbf{p}, \mathbf{x}),
\end{aligned} \tag{A.12}$$

where $x_{i,k,l}$ are numerical solution values, $\Delta^{n,m} x_{i,k,l}$ is some method-dependent finite difference discretization of $\frac{\partial^{n+m} x_i}{\partial u^n \partial v^m}$ at (u_k, v_l), and $F_{i,k,l}$ are some method-dependent functions of $\bar{F}_i$ at (u_k, v_l), for all $i \in \{1, 2, \ldots, n_x\}$ and for all $(k, l) \in \mathcal{I}_{\Delta_u} \times \mathcal{I}_{\Delta_v}$. In terms $f_{i,k,l}$ as defined in equation (A.9),

$$f_{i,k,l}(\mathbf{u}, \mathbf{v}, \mathbf{p}, \mathbf{x}) = \Delta^{1,0} x_{i,k,l} - F_{i,k,l}(\mathbf{u}, \mathbf{v}, \mathbf{p}, \mathbf{x}). \tag{A.13}$$

Thus, using a normalized least-squares measure, as described for first order ordinary differential equation models in Sections 2.3, 2.4.1, and 2.6.1,

$$\begin{aligned}
&r_{\mathrm{y}}(\mathbf{p}, \mathbf{x}) = \\
&\frac{1}{n_y} \sum_{j=1}^{n_y} \left(\frac{1}{\sum_{k=1}^{n_u} \sum_{l=1}^{n_v} w_{j,k,l} \mathrm{y}_{j,k,l}^2} \sum_{k=1}^{n_u} \sum_{l=1}^{n_v} w_{j,k,l} \big(\mathrm{y}_{j,k,l} - g_j(\mathbf{p}, x_{1,k,l}, \ldots, x_{n_x,k,l}) \big)^2 \right),
\end{aligned} \tag{A.14a}$$

$$\begin{aligned}
&r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x}) = \\
&\frac{1}{n_y} \sum_{j=1}^{n_y} \left(\frac{\hat{\sigma}}{\sum_{k \in \mathcal{I}_{\hat{\mathrm{y}}_u}} \sum_{l \in \mathcal{I}_{\hat{\mathrm{y}}_v}} \hat{w}_{j,k,l} \hat{\mathrm{y}}_{j,k,l}^2} \sum_{k \in \mathcal{I}_{\hat{\mathrm{y}}_u}} \sum_{l \in \mathcal{I}_{\hat{\mathrm{y}}_v}} \hat{w}_{j,k,l} \big(\hat{\mathrm{y}}_{j,k,l} - g_j(\mathbf{p}, x_{1,k,l}, \ldots, x_{n_x,k,l}) \big)^2 \right),
\end{aligned} \tag{A.14b}$$

$$\begin{aligned}
&r_{\Delta x}(\mathbf{p}, \mathbf{x}) = \\
&\frac{1}{n_x} \sum_{i=1}^{n_x} \left(\frac{s_i^u(\mathbf{x}) + s_i^v(\mathbf{x})}{2 \sum_{k \in \mathcal{I}_{\Delta_u}} \sum_{l \in \mathcal{I}_{\Delta_v}} (\Delta^{1,0} x_{i,k,l})^2} \sum_{k \in \mathcal{I}_{\Delta_u}} \sum_{l \in \mathcal{I}_{\Delta_v}} \big(\Delta^{1,0} x_{i,k,l} - F_{i,k,l}(\mathbf{u}, \mathbf{v}, \mathbf{p}, \mathbf{x}) \big)^2 \right),
\end{aligned} \tag{A.14c}$$

$$s_i^u(\mathbf{x}) = \alpha_i + \beta_i \left(\frac{4 \sum_{k \in \mathcal{I}_{\Delta_u} \setminus \{1, n_u\}} \sum_{l \in \mathcal{I}_{\Delta_v}} \big(x_{i,k_-,l} - 2 x_{i,k,l} + x_{i,k_+,l} \big)^2}{\sum_{k \in \mathcal{I}_{\Delta_u} \setminus \{1, n_u\}} \sum_{l \in \mathcal{I}_{\Delta_v}} \big(x_{i,k_+,l} - x_{i,k_-,l} \big)^2} \right)^{\gamma_i}, \tag{A.14d}$$

$$s_i^v(\mathbf{x}) = \alpha_i + \beta_i \left(\frac{4 \sum_{k \in \mathcal{I}_{\Delta_u}} \sum_{l \in \mathcal{I}_{\Delta_v} \setminus \{1, n_v\}} \big(x_{i,k,l_-} - 2 x_{i,k,l} + x_{i,k,l_+} \big)^2}{\sum_{k \in \mathcal{I}_{\Delta_u}} \sum_{l \in \mathcal{I}_{\Delta_v} \setminus \{1, n_v\}} \big(x_{i,k,l_+} - x_{i,k,l_-} \big)^2} \right)^{\gamma_i}, \tag{A.14e}$$

for some data-dependent weights, $w_{j,k,l}$, some interpolated data-dependent weights, $\hat{w}_{j,k,l}$, scaling parameter $\hat{\sigma}$, as discussed in Section 2.6.1, and smoothing-penalty parameters, $\alpha_i > 0$, $\beta_i \geq 0$, and $\gamma_i \geq 0$, as discussed in Section 2.4.1, where k_- and k_+ are the indices below and above k in $\mathcal{I}_{\Delta_u}$ and l_- and l_+ are the indices below and above l in $\mathcal{I}_{\Delta_v}$.

I note that under the linear indexing $m = (k, l) \in \mathcal{I}_{\Delta_u} \times \mathcal{I}_{\Delta_v} = \mathcal{I}_{\Delta}$, with $\mathbf{t} = \mathbf{u} \times \mathbf{v}$ and

$n_t = n_u n_v$, $r(\mathbf{p}, \mathbf{x}; \lambda)$ constructed from the functionals in equation (A.14) is equivalent in form to $r(\mathbf{p}, \mathbf{x}; \lambda)$ of equation (D.1). Thus, the computational complexity count from Section D.1 applies to descent on $r(\mathbf{p}, \mathbf{x}; \lambda)$ with PDEs.

Appendix B

Properties of $r(\mathbf{p}, \mathbf{x}; \lambda)$

Here, I derive properties imposed on $r(\mathbf{p}, \mathbf{x}; \lambda)$ by minimization. For $\lambda \in (0, 1)$, the parameters and state values that minimize $r(\mathbf{p}, \mathbf{x}; \lambda)$, $\breve{\mathbf{p}}^{\lambda}$ and $\breve{\mathbf{x}}^{\lambda}$, allow me to define functions, $\breve{r}(\lambda)$, $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$, as follows:

$$\begin{aligned}\breve{r}(\lambda) = (1-\lambda)\breve{r}_{\mathrm{y}}(\lambda) + (1-\lambda)^2\breve{r}_{\hat{\mathrm{y}}}(\lambda) + \lambda\breve{r}_{\Delta x}(\lambda) = \\ (1-\lambda)r_{\mathrm{y}}(\breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda}) + (1-\lambda)^2 r_{\hat{\mathrm{y}}}(\breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda}) + \lambda r_{\Delta x}(\breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda}).\end{aligned} \tag{B.1}$$

For any $\lambda \in (0, 1)$, no set of parameters and state values can further minimize $r(\mathbf{p}, \mathbf{x}; \lambda)$. Thus, for all $\lambda \in (0, 1)$ and for all ε such that $\lambda + \varepsilon \in (0, 1)$,

$$\begin{aligned}\breve{r}(\lambda) = (1-\lambda)\breve{r}_{\mathrm{y}}(\lambda) + (1-\lambda)^2\breve{r}_{\hat{\mathrm{y}}}(\lambda) + \lambda\breve{r}_{\Delta x}(\lambda) = \\ (1-\lambda)r_{\mathrm{y}}(\breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda}) + (1-\lambda)^2 r_{\hat{\mathrm{y}}}(\breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda}) + \lambda r_{\Delta x}(\breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda}) \leq \\ (1-\lambda)r_{\mathrm{y}}(\breve{\mathbf{p}}^{\lambda+\varepsilon}, \breve{\mathbf{x}}^{\lambda+\varepsilon}) + (1-\lambda)^2 r_{\hat{\mathrm{y}}}(\breve{\mathbf{p}}^{\lambda+\varepsilon}, \breve{\mathbf{x}}^{\lambda+\varepsilon}) + \lambda r_{\Delta x}(\breve{\mathbf{p}}^{\lambda+\varepsilon}, \breve{\mathbf{x}}^{\lambda+\varepsilon}) = \\ (1-\lambda)\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) + (1-\lambda)^2\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) + \lambda\breve{r}_{\Delta x}(\lambda+\varepsilon) \\ \Longleftrightarrow \\ (1-\lambda)\breve{r}_{\mathrm{y}}(\lambda) + (1-\lambda)^2\breve{r}_{\hat{\mathrm{y}}}(\lambda) + \lambda\breve{r}_{\Delta x}(\lambda) \leq \\ (1-\lambda)\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) + (1-\lambda)^2\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) + \lambda\breve{r}_{\Delta x}(\lambda+\varepsilon)\end{aligned} \tag{B.2}$$

and

$$\begin{aligned}\breve{r}(\lambda+\varepsilon) = (1-\lambda-\varepsilon)\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) + (1-\lambda-\varepsilon)^2\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) + (\lambda+\varepsilon)\breve{r}_{\Delta x}(\lambda+\varepsilon) = \\ (1-\lambda-\varepsilon)r_{\mathrm{y}}(\breve{\mathbf{p}}^{\lambda+\varepsilon}, \breve{\mathbf{x}}^{\lambda+\varepsilon}) + (1-\lambda-\varepsilon)^2 r_{\hat{\mathrm{y}}}(\breve{\mathbf{p}}^{\lambda+\varepsilon}, \breve{\mathbf{x}}^{\lambda+\varepsilon}) + (\lambda+\varepsilon) r_{\Delta x}(\breve{\mathbf{p}}^{\lambda+\varepsilon}, \breve{\mathbf{x}}^{\lambda+\varepsilon}) \leq \\ (1-\lambda-\varepsilon)r_{\mathrm{y}}(\breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda}) + (1-\lambda-\varepsilon)^2 r_{\hat{\mathrm{y}}}(\breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda}) + (\lambda+\varepsilon) r_{\Delta x}(\breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda}) = \\ (1-\lambda-\varepsilon)\breve{r}_{\mathrm{y}}(\lambda) + (1-\lambda-\varepsilon)^2\breve{r}_{\hat{\mathrm{y}}}(\lambda) + (\lambda+\varepsilon)\breve{r}_{\Delta x}(\lambda) \\ \Longleftrightarrow \\ (1-\lambda-\varepsilon)\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) + (1-\lambda-\varepsilon)^2\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) + (\lambda+\varepsilon)\breve{r}_{\Delta x}(\lambda+\varepsilon) \leq \\ (1-\lambda-\varepsilon)\breve{r}_{\mathrm{y}}(\lambda) + (1-\lambda-\varepsilon)^2\breve{r}_{\hat{\mathrm{y}}}(\lambda) + (\lambda+\varepsilon)\breve{r}_{\Delta x}(\lambda).\end{aligned} \tag{B.3}$$

From these relations, I can determine some properties of the imposed structure on $\breve{r}(\lambda)$.

B.1 Limiting Behavior of $\breve{r}(\lambda)$

Theorem 1.

$$\lim_{\lambda\to 0^+} \breve{r}(\lambda) = \lim_{\lambda\to 0^+} \min r(\mathbf{p},\mathbf{x};\lambda) = 0,$$
$$\lim_{\lambda\to 0^+} \breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda} = \arg\min \big(r_{\Delta x}(\mathbf{p},\mathbf{x}) : r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) = 0, r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) = 0\big),$$
$$\lim_{\lambda\to 1^-} \breve{r}(\lambda) = \lim_{\lambda\to 1^-} \min r(\mathbf{p},\mathbf{x};\lambda) = 0,$$
$$\lim_{\lambda\to 1^-} \breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda} = \arg\min \big(r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) : r_{\Delta x}(\mathbf{p},\mathbf{x}) = 0\big).$$

Proof. By construction, $\min\left(r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) + r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x})\right) = 0$, where observable state values match observed data and interpolated data. Thus,

$$\lim_{\lambda\to 0^+} \breve{r}(\lambda) = \lim_{\lambda\to 0^+} \min r(\mathbf{p},\mathbf{x};\lambda) =$$
$$\lim_{\lambda\to 0^+} \min \big((1-\lambda) r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) + (1-\lambda)^2 r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) + \lambda r_{\Delta x}(\mathbf{p},\mathbf{x})\big) =$$
$$\min \big(r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) + r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x})\big) = 0. \tag{B.4}$$

Thus,

$$\lim_{\lambda\to 0^+} \arg\min r(\mathbf{p},\mathbf{x};\lambda) \in \{\mathbf{p},\mathbf{x} : r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) + r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) = 0\} \implies$$
$$\lim_{\lambda\to 0^+} \arg\min r(\mathbf{p},\mathbf{x};\lambda) \in \{\mathbf{p},\mathbf{x} : r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) = 0, r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) = 0\}, \tag{B.5}$$

which implies that

$$\lim_{\lambda\to 0^+} \breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda} = \lim_{\lambda\to 0^+} \arg\min r(\mathbf{p},\mathbf{x};\lambda) =$$
$$\lim_{\lambda\to 0^+} \arg\min \big(r(\mathbf{p},\mathbf{x};\lambda) : r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) = 0, r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) = 0\big) =$$
$$\lim_{\lambda\to 0^+} \arg\min \big(\lambda r_{\Delta x}(\mathbf{p},\mathbf{x}) : r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) = 0, r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) = 0\big) =$$
$$\lim_{\lambda\to 0^+} \arg\min \big(r_{\Delta x}(\mathbf{p},\mathbf{x}) : r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) = 0, r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) = 0\big) =$$
$$\arg\min \big(r_{\Delta x}(\mathbf{p},\mathbf{x}) : r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) = 0, r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) = 0\big). \tag{B.6}$$

By construction, $\min r_{\Delta x}(\mathbf{p},\mathbf{x}) = 0$, where parameters and state values satisfy the chosen numerical solution method. Thus,

$$\lim_{\lambda\to 1^-} \breve{r}(\lambda) = \lim_{\lambda\to 1^-} \min r(\mathbf{p},\mathbf{x};\lambda) =$$
$$\lim_{\lambda\to 1^-} \min \big((1-\lambda) r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) + (1-\lambda)^2 r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) + \lambda r_{\Delta x}(\mathbf{p},\mathbf{x})\big) =$$
$$\min r_{\Delta x}(\mathbf{p},\mathbf{x}) = 0. \tag{B.7}$$

Thus,

$$\lim_{\lambda\to 1^-} \arg\min r(\mathbf{p},\mathbf{x};\lambda) \in \{\mathbf{p},\mathbf{x} : r_{\Delta x}(\mathbf{p},\mathbf{x}) = 0\}, \tag{B.8}$$

which implies that

$$\begin{gathered}
\lim_{\lambda\to 1^-} \breve{\mathbf{p}}^{\lambda}, \breve{\mathbf{x}}^{\lambda} = \lim_{\lambda\to 1^-} \arg\min r(\mathbf{p},\mathbf{x};\lambda) = \\
\lim_{\lambda\to 1^-} \arg\min \big(r(\mathbf{p},\mathbf{x};\lambda) : r_{\Delta x}(\mathbf{p},\mathbf{x}) = 0\big) = \\
\lim_{\lambda\to 1^-} \arg\min \big((1-\lambda) r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) + (1-\lambda)^2 r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) : r_{\Delta x}(\mathbf{p},\mathbf{x}) = 0\big) = \\
\lim_{\lambda\to 1^-} \arg\min \big(r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) + (1-\lambda) r_{\hat{\mathrm{y}}}(\mathbf{p},\mathbf{x}) : r_{\Delta x}(\mathbf{p},\mathbf{x}) = 0\big) = \\
\lim_{\lambda\to 1^-} \arg\min \big(r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) : r_{\Delta x}(\mathbf{p},\mathbf{x}) = 0\big) = \\
\arg\min \big(r_{\mathrm{y}}(\mathbf{p},\mathbf{x}) : r_{\Delta x}(\mathbf{p},\mathbf{x}) = 0\big).
\end{gathered} \tag{B.9}$$

□

B.2 Continuity of $\breve{r}(\lambda)$

Theorem 2. *$\breve{r}(\lambda)$ is continuous for $\lambda \in (0,1)$.*

Proof. From equation (B.3),

$$\begin{gathered}
(1-\lambda)\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) + (1-\lambda)^2 \breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) + \lambda \breve{r}_{\Delta x}(\lambda+\varepsilon) \leq \\
(1-\lambda)\breve{r}_{\mathrm{y}}(\lambda) + (1-\lambda)^2 \breve{r}_{\hat{\mathrm{y}}}(\lambda) + \lambda \breve{r}_{\Delta x}(\lambda) + \varepsilon\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + \\
2\varepsilon(1-\lambda)\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big) + \varepsilon^2\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda) - \breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon)\big) + \varepsilon\big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\Delta x}(\lambda+\varepsilon)\big),
\end{gathered} \tag{B.10}$$

which, in conjunction with equation (B.2), implies that

$$\begin{gathered}
(1-\lambda)\breve{r}_{\mathrm{y}}(\lambda) + (1-\lambda)^2 \breve{r}_{\hat{\mathrm{y}}}(\lambda) + \lambda \breve{r}_{\Delta x}(\lambda) \leq \\
(1-\lambda)\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) + (1-\lambda)^2 \breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) + \lambda \breve{r}_{\Delta x}(\lambda+\varepsilon) \leq \\
(1-\lambda)\breve{r}_{\mathrm{y}}(\lambda) + (1-\lambda)^2 \breve{r}_{\hat{\mathrm{y}}}(\lambda) + \lambda \breve{r}_{\Delta x}(\lambda) + \varepsilon\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + \\
2\varepsilon(1-\lambda)\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big) + \varepsilon^2\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda) - \breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon)\big) + \varepsilon\big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\Delta x}(\lambda+\varepsilon)\big),
\end{gathered} \tag{B.11}$$

for all $\lambda \in (0,1)$ and for all ε such that $\lambda + \varepsilon \in (0,1)$. Thus,

$$\begin{gathered}
\breve{r}(\lambda) = \lim_{\varepsilon \to 0} \big((1-\lambda)\breve{r}_{\mathrm{y}}(\lambda) + (1-\lambda)^2 \breve{r}_{\hat{\mathrm{y}}}(\lambda) + \lambda \breve{r}_{\Delta x}(\lambda)\big) \leq \\
\lim_{\varepsilon \to 0} \big((1-\lambda)\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) + (1-\lambda)^2 \breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) + \lambda \breve{r}_{\Delta x}(\lambda+\varepsilon)\big) = \\
\lim_{\varepsilon \to 0} \big((1-\lambda-\varepsilon)\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) + (1-\lambda-\varepsilon)^2 \breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) + (\lambda+\varepsilon) \breve{r}_{\Delta x}(\lambda+\varepsilon)\big) = \\
\lim_{\varepsilon \to 0} \breve{r}(\lambda+\varepsilon) \leq \\
\lim_{\varepsilon \to 0} \Big((1-\lambda)\breve{r}_{\mathrm{y}}(\lambda) + (1-\lambda)^2 \breve{r}_{\hat{\mathrm{y}}}(\lambda) + \lambda \breve{r}_{\Delta x}(\lambda) + \varepsilon\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + \\
2\varepsilon(1-\lambda)\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big) + \varepsilon^2\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda) - \breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon)\big) + \varepsilon\big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\Delta x}(\lambda+\varepsilon)\big)\Big) \\
= (1-\lambda)\breve{r}_{\mathrm{y}}(\lambda) + (1-\lambda)^2 \breve{r}_{\hat{\mathrm{y}}}(\lambda) + \lambda \breve{r}_{\Delta x}(\lambda) = \breve{r}(\lambda), \qquad \text{(B.12)}
\end{gathered}$$

which implies that

$$\lim_{\varepsilon \to 0} \breve{r}(\lambda+\varepsilon) = \breve{r}(\lambda). \qquad \text{(B.13)}$$

Therefore, $\breve{r}(\lambda)$ is continuous for $\lambda \in (0,1)$. □

B.3 Differentiability of $\breve{r}(\lambda)$

Theorem 3. *For $\lambda \in (0,1)$ such that $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ are differentiable,*

$$(1-\lambda)\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda} + (1-\lambda)^2 \frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda} + \lambda \frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda} = 0,$$

and

$$\frac{d\breve{r}(\lambda)}{d\lambda} = \breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda).$$

Proof. From Equations (B.2) and (B.3), for all $\lambda \in (0,1)$ and for all ε such that $\lambda + \varepsilon \in (0,1)$,

$$\begin{gathered}
\lambda\big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\Delta x}(\lambda+\varepsilon)\big) \leq \\
(1-\lambda)\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + (1-\lambda)^2\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big), \qquad \text{(B.14a)} \\
(1-\lambda)\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + (1-\lambda)^2\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big) \leq \\
\lambda\big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\Delta x}(\lambda+\varepsilon)\big) + \varepsilon\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + 2\varepsilon(1-\lambda)\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big) + \\
\varepsilon^2\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda) - \breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon)\big) + \varepsilon\big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\Delta x}(\lambda+\varepsilon)\big), \qquad \text{(B.14b)}
\end{gathered}$$

which implies that

$$\begin{gathered}\lambda\big(\breve{r}_{\Delta x}(\lambda)-\breve{r}_{\Delta x}(\lambda+\varepsilon)\big)\leq\\ (1-\lambda)\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon)-\breve{r}_{\mathrm{y}}(\lambda)\big)+(1-\lambda)^2\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon)-\breve{r}_{\hat{\mathrm{y}}}(\lambda)\big)\leq\\ \lambda\big(\breve{r}_{\Delta x}(\lambda)-\breve{r}_{\Delta x}(\lambda+\varepsilon)\big)+\varepsilon\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon)-\breve{r}_{\mathrm{y}}(\lambda)\big)+2\varepsilon(1-\lambda)\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon)-\breve{r}_{\hat{\mathrm{y}}}(\lambda)\big)+\\ \varepsilon^2\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda)-\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon)\big)+\varepsilon\big(\breve{r}_{\Delta x}(\lambda)-\breve{r}_{\Delta x}(\lambda+\varepsilon)\big).\end{gathered}\tag{B.15}$$

Thus, for $\varepsilon>0$,

$$\begin{gathered}\lambda\frac{\breve{r}_{\Delta x}(\lambda)-\breve{r}_{\Delta x}(\lambda+\varepsilon)}{\varepsilon}\leq\\ (1-\lambda)\frac{\breve{r}_{\mathrm{y}}(\lambda+\varepsilon)-\breve{r}_{\mathrm{y}}(\lambda)}{\varepsilon}+(1-\lambda)^2\frac{\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon)-\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{\varepsilon}\leq\\ \lambda\frac{\breve{r}_{\Delta x}(\lambda)-\breve{r}_{\Delta x}(\lambda+\varepsilon)}{\varepsilon}+\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon)-\breve{r}_{\mathrm{y}}(\lambda)\big)+2(1-\lambda)\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon)-\breve{r}_{\hat{\mathrm{y}}}(\lambda)\big)+\\ \varepsilon\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda)-\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon)\big)+\big(\breve{r}_{\Delta x}(\lambda)-\breve{r}_{\Delta x}(\lambda+\varepsilon)\big).\end{gathered}\tag{B.16}$$

For λ where $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ are differentiable, and thus also continuous, relation (B.16) implies that

$$\begin{gathered}-\lambda\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda}=\lim_{\varepsilon\to0^+}\left(\lambda\frac{\breve{r}_{\Delta x}(\lambda)-\breve{r}_{\Delta x}(\lambda+\varepsilon)}{\varepsilon}\right)\leq\\ \lim_{\varepsilon\to0^+}\left((1-\lambda)\frac{\breve{r}_{\mathrm{y}}(\lambda+\varepsilon)-\breve{r}_{\mathrm{y}}(\lambda)}{\varepsilon}+(1-\lambda)^2\frac{\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon)-\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{\varepsilon}\right)=\\ (1-\lambda)\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda}+(1-\lambda)^2\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda}\leq\\ \lim_{\varepsilon\to0^+}\left(\lambda\frac{\breve{r}_{\Delta x}(\lambda)-\breve{r}_{\Delta x}(\lambda+\varepsilon)}{\varepsilon}+\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon)-\breve{r}_{\mathrm{y}}(\lambda)\big)+2(1-\lambda)\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon)-\breve{r}_{\hat{\mathrm{y}}}(\lambda)\big)\right.\\ \left.+\varepsilon\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda)-\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon)\big)+\big(\breve{r}_{\Delta x}(\lambda)-\breve{r}_{\Delta x}(\lambda+\varepsilon)\big)\right)=-\lambda\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda},\end{gathered}\tag{B.17}$$

which implies that where $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ are differentiable,

$$-\lambda\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda}=(1-\lambda)\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda}+(1-\lambda)^2\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda}.\tag{B.18}$$

$\breve{r}(\lambda)$ is differentiable where $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ are differentiable, with value,

$$\begin{gathered}\frac{d\breve{r}(\lambda)}{d\lambda}=(1-\lambda)\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda}+(1-\lambda)^2\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda}+\lambda\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda}-\\ \breve{r}_{\mathrm{y}}(\lambda)-2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)+\breve{r}_{\Delta x}(\lambda),\end{gathered}\tag{B.19}$$

which, in conjunction with equation (B.18), implies that

$$\frac{d\breve{r}(\lambda)}{d\lambda} = \breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda). \tag{B.20}$$

□

B.4 Conservation in $\breve{r}_{\mathrm{y}}(\lambda)$

Theorem 4. *If $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ are differentiable at all but a finite number of points in $(0,1)$, then*

$$2\int_0^1 \breve{r}(\lambda)d\lambda = \int_0^1 \breve{r}_{\mathrm{y}}(\lambda)d\lambda + \int_0^1 (1-\lambda^2)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda = \int_0^1 \breve{r}_{\Delta x}(\lambda)d\lambda - \int_0^1 (1-\lambda)^2\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda.$$

Proof. On any open interval $(a,b) \in (0,1)$ over which $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ are differentiable, and thus where $\breve{r}(\lambda)$ is differentiable, from equation (B.20), the fundamental theorem of calculus, and continuity of $\breve{r}(\lambda)$ by Theorem 2,

$$\begin{aligned}
\lim_{\varepsilon\to 0^+}\left(\int_{a+\varepsilon}^{b-\varepsilon}\frac{d\breve{r}(\lambda)}{d\lambda}d\lambda\right) &= \lim_{\varepsilon\to 0^+}\left(\int_{a+\varepsilon}^{b-\varepsilon}\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda\right) \implies \\
\lim_{\varepsilon\to 0^+}\left(\breve{r}(\lambda)\Big|_{a+\varepsilon}^{b-\varepsilon}\right) &= \lim_{\varepsilon\to 0^+}\left(\int_{a+\varepsilon}^{b-\varepsilon}\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda\right) \implies \\
\breve{r}(b) - \breve{r}(a) &= \int_a^b \breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda.
\end{aligned} \tag{B.21}$$

Assuming that $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ are differentiable at all but a finite number of points, $\hat{\lambda}_1 < \hat{\lambda}_2 < \cdots < \hat{\lambda}_{\hat{n}}$, then from equation (B.21),

$$\begin{aligned}
&\lim_{\varepsilon\to 0^+}\left(\breve{r}(\hat{\lambda}_1) - \breve{r}(\varepsilon) + \sum_{i=2}^{\hat{n}}\left(\breve{r}(\hat{\lambda}_i) - \breve{r}(\hat{\lambda}_{i-1})\right) + \breve{r}(1-\varepsilon) - \breve{r}(\hat{\lambda}_{\hat{n}})\right) = \\
&\quad\lim_{\varepsilon\to 0^+}\Bigg(\int_{\varepsilon}^{\hat{\lambda}_1}\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda + \\
&\qquad\sum_{i=2}^{\hat{n}}\int_{\hat{\lambda}_{i-1}}^{\hat{\lambda}_i}\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda + \\
&\qquad\int_{\hat{\lambda}_{\hat{n}}}^{1-\varepsilon}\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda\Bigg) \implies \\
&\lim_{\varepsilon\to 0^+}\left(\breve{r}(1-\varepsilon) - \breve{r}(\varepsilon)\right) = \lim_{\varepsilon\to 0^+}\left(\int_{\varepsilon}^{1-\varepsilon}\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda\right),
\end{aligned} \tag{B.22}$$

which, in conjunction with Equations (B.4) and (B.7), imply that

$$0 = \int_0^1 \breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda \iff$$
$$\int_0^1 \breve{r}_{\mathrm{y}}(\lambda)d\lambda + 2\int_0^1 (1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda = \int_0^1 \breve{r}_{\Delta x}(\lambda)d\lambda. \tag{B.23}$$

Equation (B.23) defines the conserved quantity in the minimum deformation from data to the best fitting numerical solution. Additionally,

$$\int_0^1 \breve{r}(\lambda) = \int_0^1 (1-\lambda)\breve{r}_{\mathrm{y}}(\lambda) + (1-\lambda)^2\breve{r}_{\hat{\mathrm{y}}}(\lambda) + \lambda\breve{r}_{\Delta x}(\lambda)d\lambda =$$
$$\int_0^1 \breve{r}_{\mathrm{y}}(\lambda) + (1-\lambda^2)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda + \int_0^1 \lambda\big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\big)d\lambda. \tag{B.24}$$

On any open interval $(a,b) \in (0,1)$ over which $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ are differentiable, and thus where $\breve{r}(\lambda)$ is differentiable, from equation (B.20), integration by parts, and continuity of $\breve{r}(\lambda)$ by Theorem 2,

$$\int_a^b \lambda\big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\big)d\lambda =$$
$$\lim_{\varepsilon\to 0^+}\left(\int_{a+\varepsilon}^{b-\varepsilon} \lambda\big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\big)d\lambda\right) =$$
$$\lim_{\varepsilon\to 0^+}\left(\int_{a+\varepsilon}^{b-\varepsilon} \lambda\frac{d\breve{r}(\lambda)}{d\lambda}d\lambda\right) = \lim_{\varepsilon\to 0^+}\left(\lambda\breve{r}(\lambda)\Big|_{a+\varepsilon}^{b-\varepsilon} - \int_{a+\varepsilon}^{b-\varepsilon}\breve{r}(\lambda)d\lambda\right) =$$
$$b\breve{r}(b) - a\breve{r}(a) - \int_a^b \breve{r}(\lambda)d\lambda. \tag{B.25}$$

Assuming that $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ are differentiable, and thus $\breve{r}(\lambda)$ is differentiable, at all

but a finite number of points, $\hat{\lambda}_1 < \hat{\lambda}_2 < \cdots < \hat{\lambda}_{\hat{n}}$, then, from equation (B.25),

$$\begin{aligned}
&\int_0^1 \lambda\big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\big)d\lambda = \\
&\lim_{\varepsilon\to 0^+}\left(\int_\varepsilon^{1-\varepsilon} \lambda\big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\big)d\lambda\right) = \\
&\lim_{\varepsilon\to 0^+}\Bigg(\int_\varepsilon^{\hat{\lambda}_1} \lambda\big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\big)d\lambda + \\
&\quad\sum_{i=2}^{\hat{n}}\int_{\hat{\lambda}_{i-1}}^{\hat{\lambda}_i} \lambda\big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\big)d\lambda + \\
&\quad\int_{\hat{\lambda}_{\hat{n}}}^{1-\varepsilon} \lambda\big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda) - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\big)d\lambda\Bigg) = \\
&\lim_{\varepsilon\to 0^+}\Bigg(\hat{\lambda}_1\breve{r}(\hat{\lambda}_1) - \varepsilon\breve{r}(\varepsilon) - \int_\varepsilon^{\hat{\lambda}_1}\breve{r}(\lambda)d\lambda + \\
&\quad\sum_{i=2}^{\hat{n}}\left(\hat{\lambda}_i\breve{r}(\hat{\lambda}_i) - \hat{\lambda}_{i-1}\breve{r}(\hat{\lambda}_{i-1}) - \int_{\hat{\lambda}_{i-1}}^{\hat{\lambda}_i}\breve{r}(\lambda)d\lambda\right) + \\
&\quad(1-\varepsilon)\breve{r}(1-\varepsilon) - \hat{\lambda}_{\hat{n}}\breve{r}(\hat{\lambda}_{\hat{n}}) - \int_{\hat{\lambda}_{\hat{n}}}^{1-\varepsilon}\breve{r}(\lambda)d\lambda\Bigg) = \\
&\lim_{\varepsilon\to 0^+}\left((1-\varepsilon)\breve{r}(1-\varepsilon) - \varepsilon\breve{r}(\varepsilon) - \int_\varepsilon^{1-\varepsilon}\breve{r}(\lambda)d\lambda\right) = -\int_0^1 \breve{r}(\lambda)d\lambda,
\end{aligned} \tag{B.26}$$

as, from equation (B.7), $\lim_{\lambda\to 1^-}\breve{r}(\lambda) = 0$. Therefore, from Equations (B.24) and (B.26),

$$\begin{aligned}
\int_0^1 \breve{r}(\lambda)d\lambda = \int_0^1 \breve{r}_{\mathrm{y}}(\lambda)d\lambda + \int_0^1 (1-\lambda^2)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda - \int_0^1 \breve{r}(\lambda)d\lambda \implies \\
2\int_0^1 \breve{r}(\lambda)d\lambda = \int_0^1 \breve{r}_{\mathrm{y}}(\lambda)d\lambda + \int_0^1 (1-\lambda^2)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda,
\end{aligned} \tag{B.27}$$

which, in conjunction with equation (B.23), implies that

$$\begin{aligned}
&2\int_0^1 \breve{r}(\lambda)d\lambda = \int_0^1 \breve{r}_{\mathrm{y}}(\lambda)d\lambda + \int_0^1 (1-\lambda^2)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda = \\
&\int_0^1 \breve{r}_{\Delta x}(\lambda)d\lambda - 2\int_0^1 (1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda + \int_0^1 (1-\lambda^2)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda \iff \\
&2\int_0^1 \breve{r}(\lambda)d\lambda = \int_0^1 \breve{r}_{\mathrm{y}}(\lambda)d\lambda + \int_0^1 (1-\lambda^2)\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda = \\
&\qquad\int_0^1 \breve{r}_{\Delta x}(\lambda)d\lambda - \int_0^1 (1-\lambda)^2\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda.
\end{aligned} \tag{B.28}$$

I note, when $\breve{r}_{\hat{\mathrm{y}}}(\lambda) = 0$, equation (B.28) implies that

$$2\int_0^1 \breve{r}(\lambda)d\lambda = \int_0^1 \breve{r}_{\mathrm{y}}(\lambda)d\lambda = \int_0^1 \breve{r}_{\Delta x}(\lambda)d\lambda. \tag{B.29}$$

□

B.5 Integral Representations of Limit Values

Theorem 5. *If $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ are differentiable at all but a finite number of points in $(0,1)$, then*

$$\lim_{\lambda\to 0^+}\breve{r}_{\Delta x}(\lambda)=\int_0^1\frac{1}{\lambda^2}\breve{r}_{\mathrm{y}}(\lambda)d\lambda+\int_0^1\frac{1-\lambda^2}{\lambda^2}\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda,$$
$$\lim_{\lambda\to 1^-}\breve{r}_{\mathrm{y}}(\lambda)=\int_0^1\frac{1}{(1-\lambda)^2}\breve{r}_{\Delta x}(\lambda)d\lambda-\int_0^1\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda.$$

Proof. From equation (B.18), where $\breve{r}_{\mathrm{y}}(\lambda)$ and $\breve{r}_{\Delta x}(\lambda)$ are differentiable,

$$\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda}=-\frac{1-\lambda}{\lambda}\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda}-\frac{(1-\lambda)^2}{\lambda}\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda},\tag{B.30}$$

Assuming that $\breve{r}_{\Delta x}(\lambda)$ is differentiable at all but possibly a finite number of points, $\hat{\lambda}_1<\hat{\lambda}_2<\cdots<\hat{\lambda}_{\hat{n}}$, then, from the fundamental theorem of calculus,

$$\int_0^1\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda}d\lambda=\lim_{\varepsilon\to 0^+}\left(\int_\varepsilon^{1-\varepsilon}\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda}d\lambda\right)=$$
$$\lim_{\varepsilon\to 0^+}\left(\int_\varepsilon^{\hat{\lambda}_1-\varepsilon}\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda}d\lambda+\sum_{i=2}^{\hat{n}}\int_{\hat{\lambda}_{i-1}+\varepsilon}^{\hat{\lambda}_i-\varepsilon}\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda}d\lambda+\int_{\hat{\lambda}_{\hat{n}}+\varepsilon}^{1-\varepsilon}\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda}d\lambda\right)=$$
$$\lim_{\varepsilon\to 0^+}\left(\breve{r}_{\Delta x}(\lambda)\Big|_\varepsilon^{\hat{\lambda}_1-\varepsilon}+\sum_{i=2}^{\hat{n}}\left(\breve{r}_{\Delta x}(\lambda)\Big|_{\hat{\lambda}_{i-1}+\varepsilon}^{\hat{\lambda}_i-\varepsilon}\right)+\breve{r}_{\Delta x}(\lambda)\Big|_{\hat{\lambda}_{\hat{n}}+\varepsilon}^{1-\varepsilon}\right)=$$
$$\lim_{\varepsilon\to 0^+}\left(-\breve{r}_{\Delta x}(\varepsilon)-\sum_{i=1}^{\hat{n}}\left(\breve{r}_{\Delta x}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right)+\breve{r}_{\Delta x}(1-\varepsilon)\right),\tag{B.31}$$

which, as $\lim_{\lambda\to 1^-}\breve{r}_{\Delta x}(\lambda)=0$ (B.7), implies that

$$\int_0^1\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda}d\lambda=-\lim_{\lambda\to 0^+}\breve{r}_{\Delta x}(\lambda)-\sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left(\breve{r}_{\Delta x}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right).\tag{B.32}$$

Assuming that $\breve{r}_{\mathrm{y}}(\lambda)$ is differentiable at all but possibly a finite number of points, $\hat{\lambda}_1<\hat{\lambda}_2<$

$\cdots < \hat{\lambda}_{\hat{n}}$, then, from integration by parts,

$$
\begin{aligned}
-\int_0^1 \frac{1-\lambda}{\lambda}\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda}d\lambda = -\lim_{\varepsilon\to 0^+}\left(\int_\varepsilon^{1-\varepsilon}\frac{1-\lambda}{\lambda}\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda}d\lambda\right) = \\
-\lim_{\varepsilon\to 0^+}\left(\int_\varepsilon^{\hat{\lambda}_1-\varepsilon}\frac{1-\lambda}{\lambda}\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda}d\lambda + \sum_{i=2}^{\hat{n}}\int_{\hat{\lambda}_{i-1}+\varepsilon}^{\hat{\lambda}_i-\varepsilon}\frac{1-\lambda}{\lambda}\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda}d\lambda + \right. \\
\left.\int_{\hat{\lambda}_{\hat{n}}+\varepsilon}^{1-\varepsilon}\frac{1-\lambda}{\lambda}\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda}d\lambda\right) = \\
-\lim_{\varepsilon\to 0^+}\left(\frac{1-\lambda}{\lambda}\breve{r}_{\mathrm{y}}(\lambda)\Big|_\varepsilon^{\hat{\lambda}_1-\varepsilon} + \int_\varepsilon^{\hat{\lambda}_1-\varepsilon}\frac{1}{\lambda^2}\breve{r}_{\mathrm{y}}(\lambda)d\lambda + \right. \\
\sum_{i=2}^{\hat{n}}\left(\frac{1-\lambda}{\lambda}\breve{r}_{\mathrm{y}}(\lambda)\Big|_{\hat{\lambda}_{i-1}+\varepsilon}^{\hat{\lambda}_i-\varepsilon} + \int_{\hat{\lambda}_{i-1}+\varepsilon}^{\hat{\lambda}_i-\varepsilon}\frac{1}{\lambda^2}\breve{r}_{\mathrm{y}}(\lambda)d\lambda\right) + \\
\left.\frac{1-\lambda}{\lambda}\breve{r}_{\mathrm{y}}(\lambda)\Big|_{\hat{\lambda}_{\hat{n}}+\varepsilon}^{1-\varepsilon} + \int_{\hat{\lambda}_{\hat{n}}+\varepsilon}^{1-\varepsilon}\frac{1}{\lambda^2}\breve{r}_{\mathrm{y}}(\lambda)d\lambda\right) = \\
\lim_{\varepsilon\to 0^+}\left(\frac{1-\varepsilon}{\varepsilon}\breve{r}_{\mathrm{y}}(\varepsilon)\right) - \lim_{\varepsilon\to 0^+}\left(\frac{\varepsilon}{1-\varepsilon}\breve{r}_{\mathrm{y}}(1-\varepsilon)\right) + \\
\sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left(\frac{1-\lambda}{\lambda}\breve{r}_{\mathrm{y}}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right) - \int_0^1\frac{1}{\lambda^2}\breve{r}_{\mathrm{y}}(\lambda)d\lambda = \\
\lim_{\varepsilon\to 0^+}\left(\frac{1-\varepsilon}{\varepsilon}\breve{r}_{\mathrm{y}}(\varepsilon)\right) + \sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left(\frac{1-\lambda}{\lambda}\breve{r}_{\mathrm{y}}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right) - \int_0^1\frac{1}{\lambda^2}\breve{r}_{\mathrm{y}}(\lambda)d\lambda. \qquad \text{(B.33)}
\end{aligned}
$$

As $\lim_{\lambda\to 0^+}\breve{r}_{\mathrm{y}}(\lambda) = 0$ (B.4), from L'Hôpital's rule,

$$
\begin{aligned}
\lim_{\varepsilon\to 0^+}\left(\frac{1-\varepsilon}{\varepsilon}\breve{r}_{\mathrm{y}}(\varepsilon)\right) = \lim_{\lambda\to 0^+}\left(\frac{1-\lambda}{\lambda}\breve{r}_{\mathrm{y}}(\lambda)\right) = \lim_{\lambda\to 0^+}\left((1-\lambda)\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda} - \breve{r}_{\mathrm{y}}(\lambda)\right) = \\
\lim_{\lambda\to 0^+}\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda}. \qquad \text{(B.34)}
\end{aligned}
$$

From equation (B.18),

$$
\begin{aligned}
0 = \lim_{\lambda\to 0^+}\left(-\lambda\frac{d\breve{\Delta x}(\lambda)}{d\lambda}\right) = \lim_{\lambda\to 0^+}\left((1-\lambda)\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda} + (1-\lambda)^2\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda}\right) = \\
\lim_{\lambda\to 0^+}\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda} + \lim_{\lambda\to 0^+}\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda}. \qquad \text{(B.35)}
\end{aligned}
$$

By construction, $\breve{r}_{\mathrm{y}}(\lambda) \geq 0$ and $\breve{r}_{\hat{\mathrm{y}}}(\lambda) \geq 0$, $\forall\lambda \in (0,1)$; from equation (B.4), $\lim_{\lambda\to 0^+}\breve{r}_{\mathrm{y}}(\lambda) = 0$ and $\lim_{\lambda\to 0^+}\breve{r}_{\hat{\mathrm{y}}}(\lambda) = 0$. Thus, by L'Hôpital's rule,

$$
0 \leq \frac{\breve{r}_{\mathrm{y}}(\lambda)}{\lambda} \implies 0 \leq \lim_{\lambda\to 0^+}\frac{\breve{r}_{\mathrm{y}}(\lambda)}{\lambda} = \lim_{\lambda\to 0^+}\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda} \qquad \text{(B.36a)}
$$

$$
0 \leq \frac{\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{\lambda} \implies 0 \leq \lim_{\lambda\to 0^+}\frac{\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{\lambda} = \lim_{\lambda\to 0^+}\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda}, \qquad \text{(B.36b)}
$$

which, in conjunction with equation (B.35), implies that

$$\lim_{\lambda\to 0^+} \frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda} = 0 \tag{B.37a}$$

$$\lim_{\lambda\to 0^+} \frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda} = 0. \tag{B.37b}$$

Equations (B.34) and (B.37a) imply that

$$\lim_{\varepsilon\to 0^+} \left(\frac{1-\varepsilon}{\varepsilon} \breve{r}_{\mathrm{y}}(\varepsilon) \right) = 0, \tag{B.38}$$

which, in conjunction with equation (B.33), implies that

$$-\int_0^1 \frac{1-\lambda}{\lambda} \frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda} d\lambda = \sum_{i=1}^{\hat{n}} \lim_{\varepsilon\to 0^+} \left(\frac{1-\lambda}{\lambda} \breve{r}_{\mathrm{y}}(\lambda) \Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon} \right) - \int_0^1 \frac{1}{\lambda^2} \breve{r}_{\mathrm{y}}(\lambda) d\lambda. \tag{B.39}$$

Assuming that $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$ is differentiable at all but possibly a finite number of points, $\hat{\lambda}_1 < \hat{\lambda}_2 < \cdots < \hat{\lambda}_{\hat{n}}$, then, from integration by parts,

$$\begin{aligned}
&-\int_0^1 \frac{(1-\lambda)^2}{\lambda} \frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda} d\lambda = -\lim_{\varepsilon\to 0^+} \left(\int_\varepsilon^{1-\varepsilon} \frac{(1-\lambda)^2}{\lambda} \frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda} d\lambda \right) = \\
&-\lim_{\varepsilon\to 0^+} \left(\int_\varepsilon^{\hat{\lambda}_1-\varepsilon} \frac{(1-\lambda)^2}{\lambda} \frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda} d\lambda + \sum_{i=2}^{\hat{n}} \int_{\hat{\lambda}_{i-1}+\varepsilon}^{\hat{\lambda}_i-\varepsilon} \frac{(1-\lambda)^2}{\lambda} \frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda} d\lambda + \right. \\
&\qquad \left. \int_{\hat{\lambda}_{\hat{n}}+\varepsilon}^{1-\varepsilon} \frac{(1-\lambda)^2}{\lambda} \frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda} d\lambda \right) = \\
&-\lim_{\varepsilon\to 0^+} \left(\frac{(1-\lambda)^2}{\lambda} \breve{r}_{\hat{\mathrm{y}}}(\lambda) \Big|_\varepsilon^{\hat{\lambda}_1-\varepsilon} + \int_\varepsilon^{\hat{\lambda}_1-\varepsilon} \frac{1-\lambda^2}{\lambda^2} \breve{r}_{\hat{\mathrm{y}}}(\lambda) d\lambda + \right. \\
&\sum_{i=2}^{\hat{n}} \left(\frac{(1-\lambda)^2}{\lambda} \breve{r}_{\hat{\mathrm{y}}}(\lambda) \Big|_{\hat{\lambda}_{i-1}+\varepsilon}^{\hat{\lambda}_i-\varepsilon} + \int_{\hat{\lambda}_{i-1}+\varepsilon}^{\hat{\lambda}_i-\varepsilon} \frac{1-\lambda^2}{\lambda^2} \breve{r}_{\hat{\mathrm{y}}}(\lambda) d\lambda \right) + \\
&\qquad \left. \frac{(1-\lambda)^2}{\lambda} \breve{r}_{\hat{\mathrm{y}}}(\lambda) \Big|_{\hat{\lambda}_{\hat{n}}+\varepsilon}^{1-\varepsilon} + \int_{\hat{\lambda}_{\hat{n}}+\varepsilon}^{1-\varepsilon} \frac{1-\lambda^2}{\lambda^2} \breve{r}_{\hat{\mathrm{y}}}(\lambda) d\lambda \right) = \\
&\lim_{\varepsilon\to 0^+} \left(\frac{(1-\varepsilon)^2}{\varepsilon} \breve{r}_{\hat{\mathrm{y}}}(\varepsilon) \right) - \lim_{\varepsilon\to 0^+} \left(\frac{\varepsilon^2}{1-\varepsilon} \breve{r}_{\hat{\mathrm{y}}}(1-\varepsilon) \right) + \\
&\sum_{i=1}^{\hat{n}} \lim_{\varepsilon\to 0^+} \left(\frac{(1-\lambda)^2}{\lambda} \breve{r}_{\hat{\mathrm{y}}}(\lambda) \Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon} \right) - \int_0^1 \frac{1-\lambda^2}{\lambda^2} \breve{r}_{\hat{\mathrm{y}}}(\lambda) d\lambda = \\
&\lim_{\varepsilon\to 0^+} \left(\frac{(1-\varepsilon)^2}{\varepsilon} \breve{r}_{\hat{\mathrm{y}}}(\varepsilon) \right) + \sum_{i=1}^{\hat{n}} \lim_{\varepsilon\to 0^+} \left(\frac{(1-\lambda)^2}{\lambda} \breve{r}_{\hat{\mathrm{y}}}(\lambda) \Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon} \right) - \int_0^1 \frac{1-\lambda^2}{\lambda^2} \breve{r}_{\hat{\mathrm{y}}}(\lambda) d\lambda.
\end{aligned} \tag{B.40}$$

From L'Hôpital's rule, as $\lim_{\lambda\to 0^+} \breve{r}_{\hat{\mathrm{y}}}(\lambda) = 0$ (B.4), and from equation (B.37b),

$$\lim_{\varepsilon\to 0^+}\left(\frac{(1-\varepsilon)^2}{\varepsilon}\breve{r}_{\hat{\mathrm{y}}}(\varepsilon)\right) = \lim_{\lambda\to 0^+}\left(\frac{(1-\lambda)^2}{\lambda}\breve{r}_{\hat{\mathrm{y}}}(\lambda)\right) = \lim_{\lambda\to 0^+}\left((1-\lambda)^2\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda} - 2(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\right) = \lim_{\lambda\to 0^+}\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda} = 0, \tag{B.41}$$

which, in conjunction with equation (B.40), implies that

$$-\int_0^1 \frac{(1-\lambda)^2}{\lambda}\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda}d\lambda = \sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left(\frac{(1-\lambda)^2}{\lambda}\breve{r}_{\hat{\mathrm{y}}}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right) - \int_0^1 \frac{1-\lambda^2}{\lambda^2}\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda. \tag{B.42}$$

From equation (B.15), for all $\lambda \in (0,1)$ and for all ε such that $\lambda+\varepsilon \in (0,1)$,

$$\begin{aligned}
&0 \leq (1-\lambda)\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + \\
&(1-\lambda)^2\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big) + \lambda\big(\breve{r}_{\Delta x}(\lambda+\varepsilon) - \breve{r}_{\Delta x}(\lambda)\big) \leq \\
&\varepsilon\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + 2\varepsilon(1-\lambda)\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big) + \\
&\varepsilon^2\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda) - \breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon)\big) + \varepsilon\big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\Delta x}(\lambda+\varepsilon)\big).
\end{aligned} \tag{B.43}$$

Thus, for all $\lambda \in (0,1)$,

$$\begin{aligned}
&0 = \lim_{\varepsilon\to 0} 0 \leq \lim_{\varepsilon\to 0}\Big((1-\lambda)\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + \\
&(1-\lambda)^2\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big) + \lambda\big(\breve{r}_{\Delta x}(\lambda+\varepsilon) - \breve{r}_{\Delta x}(\lambda)\big)\Big) \leq \\
&\lim_{\varepsilon\to 0}\Big(\varepsilon\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + 2\varepsilon(1-\lambda)\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big) + \\
&\varepsilon^2\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda) - \breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon)\big) + \varepsilon\big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\Delta x}(\lambda+\varepsilon)\big)\Big) = 0 \implies \\
&\lim_{\varepsilon\to 0}\Big((1-\lambda)\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + (1-\lambda)^2\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big) + \\
&\lambda\big(\breve{r}_{\Delta x}(\lambda+\varepsilon) - \breve{r}_{\Delta x}(\lambda)\big)\Big) = 0 \iff \\
&\lim_{\varepsilon\to 0^\pm}\Big((1-\lambda)\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + (1-\lambda)^2\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big) + \\
&\lambda\big(\breve{r}_{\Delta x}(\lambda+\varepsilon) - \breve{r}_{\Delta x}(\lambda)\big)\Big) = 0 \implies \\
&(1-\lambda)\lim_{\varepsilon\to 0^\pm}\big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + (1-\lambda)^2\lim_{\varepsilon\to 0^\pm}\big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big) + \\
&\lambda\lim_{\varepsilon\to 0^\pm}\big(\breve{r}_{\Delta x}(\lambda+\varepsilon) - \breve{r}_{\Delta x}(\lambda)\big) = 0,
\end{aligned} \tag{B.44}$$

and consequently, for all $\lambda \in (0,1)$,

$$
\begin{gathered}
(1-\lambda) \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda-\varepsilon)\big) + \\
(1-\lambda)^2 \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda-\varepsilon)\big) + \\
\lambda \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\Delta x}(\lambda+\varepsilon) - \breve{r}_{\Delta x}(\lambda-\varepsilon)\big) = \\
(1-\lambda) \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda) + \breve{r}_{\mathrm{y}}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda-\varepsilon)\big) + \\
(1-\lambda)^2 \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda) + \breve{r}_{\hat{\mathrm{y}}}(\lambda) - \breve{r}_{\hat{\mathrm{y}}}(\lambda-\varepsilon)\big) + \\
\lambda \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\Delta x}(\lambda+\varepsilon) - \breve{r}_{\Delta x}(\lambda) + \breve{r}_{\Delta x}(\lambda) - \breve{r}_{\Delta x}(\lambda-\varepsilon)\big) = \\
(1-\lambda) \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + (1-\lambda) \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\mathrm{y}}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda-\varepsilon)\big) + \\
(1-\lambda)^2 \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big) + (1-\lambda)^2 \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\hat{\mathrm{y}}}(\lambda) - \breve{r}_{\hat{\mathrm{y}}}(\lambda-\varepsilon)\big) + \\
\lambda \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\Delta x}(\lambda+\varepsilon) - \breve{r}_{\Delta x}(\lambda)\big) + \lambda \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\Delta x}(\lambda-\varepsilon)\big) = \\
(1-\lambda) \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + (1-\lambda) \lim_{\varepsilon \to 0^-} \big(\breve{r}_{\mathrm{y}}(\lambda) - \breve{r}_{\mathrm{y}}(\lambda+\varepsilon)\big) + \\
(1-\lambda)^2 \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big) + (1-\lambda)^2 \lim_{\varepsilon \to 0^-} \big(\breve{r}_{\hat{\mathrm{y}}}(\lambda) - \breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon)\big) + \\
\lambda \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\Delta x}(\lambda+\varepsilon) - \breve{r}_{\Delta x}(\lambda)\big) + \lambda \lim_{\varepsilon \to 0^-} \big(\breve{r}_{\Delta x}(\lambda) - \breve{r}_{\Delta x}(\lambda+\varepsilon)\big) = \\
\Big((1-\lambda) \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + (1-\lambda)^2 \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big) + \\
\lambda \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\Delta x}(\lambda+\varepsilon) - \breve{r}_{\Delta x}(\lambda)\big)\Big) - \\
\Big((1-\lambda) \lim_{\varepsilon \to 0^-} \big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda)\big) + (1-\lambda)^2 \lim_{\varepsilon \to 0^-} \big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda)\big) + \\
\lambda \lim_{\varepsilon \to 0^-} \big(\breve{r}_{\Delta x}(\lambda+\varepsilon) - \breve{r}_{\Delta x}(\lambda)\big)\Big) = 0 - 0 = 0 \iff \\
(1-\lambda) \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\mathrm{y}}(\lambda+\varepsilon) - \breve{r}_{\mathrm{y}}(\lambda-\varepsilon)\big) + (1-\lambda)^2 \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\hat{\mathrm{y}}}(\lambda+\varepsilon) - \breve{r}_{\hat{\mathrm{y}}}(\lambda-\varepsilon)\big) + \\
\lambda \lim_{\varepsilon \to 0^+} \big(\breve{r}_{\Delta x}(\lambda+\varepsilon) - \breve{r}_{\Delta x}(\lambda-\varepsilon)\big) = 0.
\end{gathered}
\tag{B.45}
$$

Assuming that $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ are differentiable at all but a finite number of points, $\hat{\lambda}_1 < \hat{\lambda}_2 < \cdots < \hat{\lambda}_{\hat{n}}$, then, from equation (B.30),

$$
\int_0^1 \frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda} d\lambda = -\int_0^1 \frac{1-\lambda}{\lambda} \frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda} d\lambda - \int_0^1 \frac{(1-\lambda)^2}{\lambda} \frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda} d\lambda, \tag{B.46}
$$

which, in conjunction with Equations (B.32), (B.39), and (B.42) implies that

$$
\begin{aligned}
-\lim_{\lambda\to 0^+}\breve{r}_{\Delta x}(\lambda) - \sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left(\breve{r}_{\Delta x}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right) = \\
\sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left(\frac{1-\lambda}{\lambda}\breve{r}_{\mathrm{y}}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right) - \int_0^1 \frac{1}{\lambda^2}\breve{r}_{\mathrm{y}}(\lambda)d\lambda + \\
\sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left(\frac{(1-\lambda)^2}{\lambda}\breve{r}_{\hat{\mathrm{y}}}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right) - \int_0^1 \frac{1-\lambda^2}{\lambda^2}\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda.
\end{aligned} \tag{B.47}
$$

From equation (B.45), for $\hat{\lambda}\in(0,1)$,

$$
\begin{aligned}
\lim_{\varepsilon\to 0^+}\left(\frac{1-\lambda}{\lambda}\breve{r}_{\mathrm{y}}(\lambda)\Big|_{\hat{\lambda}-\varepsilon}^{\hat{\lambda}+\varepsilon}\right) + \lim_{\varepsilon\to 0^+}\left(\frac{(1-\lambda)^2}{\lambda}\breve{r}_{\hat{\mathrm{y}}}(\lambda)\Big|_{\hat{\lambda}-\varepsilon}^{\hat{\lambda}+\varepsilon}\right) = \\
\frac{1-\hat{\lambda}}{\hat{\lambda}}\lim_{\varepsilon\to 0^+}\left(\breve{r}_{\mathrm{y}}(\lambda)\Big|_{\hat{\lambda}-\varepsilon}^{\hat{\lambda}+\varepsilon}\right) + \frac{(1-\hat{\lambda})^2}{\hat{\lambda}}\lim_{\varepsilon\to 0^+}\left(\breve{r}_{\hat{\mathrm{y}}}(\lambda)\Big|_{\hat{\lambda}-\varepsilon}^{\hat{\lambda}+\varepsilon}\right) = -\lim_{\varepsilon\to 0^+}\left(\breve{r}_{\Delta x}(\lambda)\Big|_{\hat{\lambda}-\varepsilon}^{\hat{\lambda}+\varepsilon}\right),
\end{aligned} \tag{B.48}
$$

which, in conjunction with equation (B.47), implies that

$$
\begin{aligned}
-\lim_{\lambda\to 0^+}\breve{r}_{\Delta x}(\lambda) - \sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left(\breve{r}_{\Delta x}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right) = \\
-\sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left(\breve{r}_{\Delta x}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right) - \int_0^1 \frac{1}{\lambda^2}\breve{r}_{\mathrm{y}}(\lambda)d\lambda - \int_0^1 \frac{1-\lambda^2}{\lambda^2}\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda \implies \\
\lim_{\lambda\to 0^+}\breve{r}_{\Delta x}(\lambda) = \int_0^1 \frac{1}{\lambda^2}\breve{r}_{\mathrm{y}}(\lambda)d\lambda + \int_0^1 \frac{1-\lambda^2}{\lambda^2}\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda.
\end{aligned} \tag{B.49}
$$

Similarly, from equation (B.18), where $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ are differentiable,

$$
\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda} = -\frac{\lambda}{1-\lambda}\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda} - (1-\lambda)\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda}. \tag{B.50}
$$

Assuming that $\breve{r}_{\mathrm{y}}(\lambda)$ is differentiable at all but possibly a finite number of points, $\hat{\lambda}_1 < \hat{\lambda}_2 < \cdots < \hat{\lambda}_{\hat{n}}$, then, from the fundamental theorem of calculus,

$$
\begin{aligned}
\int_0^1 \frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda}d\lambda = \lim_{\varepsilon\to 0^+}\left(\int_\varepsilon^{1-\varepsilon}\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda}d\lambda\right) = \\
\lim_{\varepsilon\to 0^+}\left(\int_\varepsilon^{\hat{\lambda}_1-\varepsilon}\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda}d\lambda + \sum_{i=2}^{\hat{n}}\int_{\hat{\lambda}_{i-1}+\varepsilon}^{\hat{\lambda}_i-\varepsilon}\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda}d\lambda + \int_{\hat{\lambda}_{\hat{n}}+\varepsilon}^{1-\varepsilon}\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda}d\lambda\right) = \\
\lim_{\varepsilon\to 0^+}\left(\breve{r}_{\mathrm{y}}(\lambda)\Big|_{\varepsilon}^{\hat{\lambda}_1-\varepsilon} + \sum_{i=2}^{\hat{n}}\left(\breve{r}_{\mathrm{y}}(\lambda)\Big|_{\hat{\lambda}_{i-1}+\varepsilon}^{\hat{\lambda}_i-\varepsilon}\right) + \breve{r}_{\mathrm{y}}(\lambda)\Big|_{\hat{\lambda}_{\hat{n}}+\varepsilon}^{1-\varepsilon}\right) = \\
\lim_{\varepsilon\to 0^+}\left(-\breve{r}_{\mathrm{y}}(\varepsilon) - \sum_{i=1}^{\hat{n}}\left(\breve{r}_{\mathrm{y}}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right) + \breve{r}_{\mathrm{y}}(1-\varepsilon)\right),
\end{aligned} \tag{B.51}
$$

which, as $\lim_{\lambda\to 0^+} \breve{r}_{\mathrm{y}}(\lambda) = 0$ (B.4), implies that

$$\int_0^1 \frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda} d\lambda = \lim_{\lambda\to 1^-} \breve{r}_{\mathrm{y}}(\lambda) - \sum_{i=1}^{\hat{n}} \lim_{\varepsilon\to 0^+} \left(\breve{r}_{\mathrm{y}}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon} \right). \tag{B.52}$$

Assuming that $\breve{r}_{\Delta x}(\lambda)$ is differentiable at all but possibly a finite number of points, $\hat{\lambda}_1 < \hat{\lambda}_2 < \cdots < \hat{\lambda}_{\hat{n}}$, then, from integration by parts,

$$\begin{aligned}
-\int_0^1 \frac{\lambda}{1-\lambda}\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda} d\lambda = -\lim_{\varepsilon\to 0^+} \left(\int_\varepsilon^{1-\varepsilon} \frac{\lambda}{1-\lambda}\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda} d\lambda \right) = \\
-\lim_{\varepsilon\to 0^+} \left(\int_\varepsilon^{\hat{\lambda}_1-\varepsilon} \frac{\lambda}{1-\lambda}\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda} d\lambda + \sum_{i=2}^{\hat{n}} \int_{\hat{\lambda}_{i-1}+\varepsilon}^{\hat{\lambda}_i-\varepsilon} \frac{\lambda}{1-\lambda}\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda} d\lambda + \right. \\
\left. \int_{\hat{\lambda}_{\hat{n}}+\varepsilon}^{1-\varepsilon} \frac{\lambda}{1-\lambda}\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda} d\lambda \right) = \\
-\lim_{\varepsilon\to 0^+} \left(\frac{\lambda}{1-\lambda}\breve{r}_{\Delta x}(\lambda)\Big|_\varepsilon^{\hat{\lambda}_1-\varepsilon} - \int_\varepsilon^{\hat{\lambda}_1-\varepsilon} \frac{1}{(1-\lambda)^2}\breve{r}_{\Delta x}(\lambda) d\lambda + \right. \\
\sum_{i=2}^{\hat{n}} \left(\frac{\lambda}{1-\lambda}\breve{r}_{\Delta x}(\lambda)\Big|_{\hat{\lambda}_{i-1}+\varepsilon}^{\hat{\lambda}_i-\varepsilon} - \int_{\hat{\lambda}_{i-1}+\varepsilon}^{\hat{\lambda}_i-\varepsilon} \frac{1}{(1-\lambda)^2}\breve{r}_{\Delta x}(\lambda) d\lambda \right) + \\
\left. \frac{\lambda}{1-\lambda}\breve{r}_{\Delta x}(\lambda)\Big|_{\hat{\lambda}_{\hat{n}}+\varepsilon}^{1-\varepsilon} - \int_{\hat{\lambda}_{\hat{n}}+\varepsilon}^{1-\varepsilon} \frac{1}{(1-\lambda)^2}\breve{r}_{\Delta x}(\lambda) d\lambda \right) = \\
\lim_{\varepsilon\to 0^+} \left(\frac{\varepsilon}{1-\varepsilon}\breve{r}_{\Delta x}(\varepsilon) \right) - \lim_{\varepsilon\to 0^+} \left(\frac{1-\varepsilon}{\varepsilon}\breve{r}_{\Delta x}(1-\varepsilon) \right) + \\
\sum_{i=1}^{\hat{n}} \lim_{\varepsilon\to 0^+} \left(\frac{\lambda}{1-\lambda}\breve{r}_{\Delta x}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon} \right) + \int_0^1 \frac{1}{(1-\lambda)^2}\breve{r}_{\Delta x}(\lambda) d\lambda = \\
-\lim_{\varepsilon\to 0^+} \left(\frac{1-\varepsilon}{\varepsilon}\breve{r}_{\Delta x}(1-\varepsilon) \right) + \sum_{i=1}^{\hat{n}} \lim_{\varepsilon\to 0^+} \left(\frac{\lambda}{1-\lambda}\breve{r}_{\Delta x}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon} \right) + \int_0^1 \frac{1}{(1-\lambda)^2}\breve{r}_{\Delta x}(\lambda) d\lambda.
\end{aligned} \tag{B.53}$$

As $\lim_{\lambda\to 1^-} \breve{r}_{\Delta x}(\lambda) = 0$ (B.7), from L'Hôpital's rule,

$$\begin{aligned}
\lim_{\varepsilon\to 0^+} \left(\frac{1-\varepsilon}{\varepsilon}\breve{r}_{\Delta x}(1-\varepsilon) \right) = \lim_{\lambda\to 1^-} \left(\frac{\lambda}{1-\lambda}\breve{r}_{\Delta x}(\lambda) \right) = \lim_{\lambda\to 1^-} \left(-\lambda\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda} - \breve{r}_{\Delta x}(\lambda) \right) = \\
-\lim_{\lambda\to 1^-} \frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda}.
\end{aligned} \tag{B.54}$$

From equation (B.18),

$$\begin{aligned}
-\lim_{\lambda\to 1^-} \frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda} = \lim_{\lambda\to 1^-} \left(-\lambda\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda} \right) = \\
\lim_{\lambda\to 1^-} \left((1-\lambda)\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda} + (1-\lambda)^2\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda} \right) = 0,
\end{aligned} \tag{B.55}$$

which, in conjunction with equation (B.54), implies that

$$\lim_{\varepsilon\to 0^+}\left(\frac{1-\varepsilon}{\varepsilon}\breve{r}_{\Delta x}(1-\varepsilon)\right)=0, \tag{B.56}$$

which, in conjunction with equation (B.53), implies that

$$-\int_0^1\frac{\lambda}{1-\lambda}\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda}d\lambda=\sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left(\frac{\lambda}{1-\lambda}\breve{r}_{\Delta x}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right)+\int_0^1\frac{1}{(1-\lambda)^2}\breve{r}_{\Delta x}(\lambda)d\lambda. \tag{B.57}$$

Assuming that $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$ is differentiable at all but possibly a finite number of points, $\hat{\lambda}_1<\hat{\lambda}_2<\cdots<\hat{\lambda}_{\hat{n}}$, then, from integration by parts,

$$\begin{aligned}
-\int_0^1(1-\lambda)\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda}d\lambda&=-\lim_{\varepsilon\to 0^+}\left(\int_\varepsilon^{1-\varepsilon}(1-\lambda)\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda}d\lambda\right)=\\
-\lim_{\varepsilon\to 0^+}&\left(\int_\varepsilon^{\hat{\lambda}_1-\varepsilon}(1-\lambda)\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda}d\lambda+\sum_{i=2}^{\hat{n}}\int_{\hat{\lambda}_{i-1}+\varepsilon}^{\hat{\lambda}_i-\varepsilon}(1-\lambda)\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda}d\lambda+\right.\\
&\left.\int_{\hat{\lambda}_{\hat{n}}+\varepsilon}^{1-\varepsilon}(1-\lambda)\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda}d\lambda\right)=\\
-\lim_{\varepsilon\to 0^+}&\left((1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\Big|_\varepsilon^{\hat{\lambda}_1-\varepsilon}+\int_\varepsilon^{\hat{\lambda}_1-\varepsilon}\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda+\right.\\
&\sum_{i=2}^{\hat{n}}\left((1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\Big|_{\hat{\lambda}_{i-1}+\varepsilon}^{\hat{\lambda}_i-\varepsilon}+\int_{\hat{\lambda}_{i-1}+\varepsilon}^{\hat{\lambda}_i-\varepsilon}\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda\right)+\\
&\left.(1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\Big|_{\hat{\lambda}_{\hat{n}}+\varepsilon}^{1-\varepsilon}+\int_{\hat{\lambda}_{\hat{n}}+\varepsilon}^{1-\varepsilon}\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda\right)=\\
&\lim_{\varepsilon\to 0^+}\left((1-\varepsilon)\breve{r}_{\hat{\mathrm{y}}}(\varepsilon)\right)-\lim_{\varepsilon\to 0^+}\left(\varepsilon\breve{r}_{\hat{\mathrm{y}}}(1-\varepsilon)\right)+\\
&\sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left((1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right)-\int_0^1\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda=\\
&\lim_{\varepsilon\to 0^+}\breve{r}_{\hat{\mathrm{y}}}(\varepsilon)+\sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left((1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right)-\int_0^1\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda,
\end{aligned} \tag{B.58}$$

which, as $\lim_{\lambda\to 0^+}\breve{r}_{\hat{\mathrm{y}}}(\lambda)=0$ (B.4), implies that

$$-\int_0^1(1-\lambda)\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda}d\lambda=\sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left((1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right)-\int_0^1\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda. \tag{B.59}$$

Assuming that $\breve{r}_{\mathrm{y}}(\lambda)$, $\breve{r}_{\hat{\mathrm{y}}}(\lambda)$, and $\breve{r}_{\Delta x}(\lambda)$ are differentiable at all but a finite number of points, $\hat{\lambda}_1<\hat{\lambda}_2<\cdots<\hat{\lambda}_{\hat{n}}$, then, from equation (B.50),

$$\int_0^1\frac{d\breve{r}_{\mathrm{y}}(\lambda)}{d\lambda}d\lambda=-\int_0^1\frac{\lambda}{1-\lambda}\frac{d\breve{r}_{\Delta x}(\lambda)}{d\lambda}d\lambda-\int_0^1(1-\lambda)\frac{d\breve{r}_{\hat{\mathrm{y}}}(\lambda)}{d\lambda}d\lambda, \tag{B.60}$$

which, in conjunction with Equations (B.52), (B.57), and (B.59) implies that

$$\lim_{\lambda\to 1^-}\breve{r}_{\mathrm{y}}(\lambda)-\sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left(\breve{r}_{\mathrm{y}}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right)=$$
$$\sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left(\frac{\lambda}{1-\lambda}\breve{r}_{\Delta x}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right)+\int_0^1\frac{1}{(1-\lambda)^2}\breve{r}_{\Delta x}(\lambda)d\lambda+$$
$$\sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left((1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right)-\int_0^1\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda. \tag{B.61}$$

From equation (B.45), for $\hat{\lambda}\in(0,1)$,

$$\lim_{\varepsilon\to 0^+}\left(\frac{\lambda}{1-\lambda}\breve{r}_{\Delta x}(\lambda)\Big|_{\hat{\lambda}-\varepsilon}^{\hat{\lambda}+\varepsilon}\right)+\lim_{\varepsilon\to 0^+}\left((1-\lambda)\breve{r}_{\hat{\mathrm{y}}}(\lambda)\Big|_{\hat{\lambda}-\varepsilon}^{\hat{\lambda}+\varepsilon}\right)=$$
$$\frac{\hat{\lambda}}{1-\hat{\lambda}}\lim_{\varepsilon\to 0^+}\left(\breve{r}_{\Delta x}(\lambda)\Big|_{\hat{\lambda}-\varepsilon}^{\hat{\lambda}+\varepsilon}\right)+(1-\hat{\lambda})\lim_{\varepsilon\to 0^+}\left(\breve{r}_{\hat{\mathrm{y}}}(\lambda)\Big|_{\hat{\lambda}-\varepsilon}^{\hat{\lambda}+\varepsilon}\right)=-\lim_{\varepsilon\to 0^+}\left(\breve{r}_{\mathrm{y}}(\lambda)\Big|_{\hat{\lambda}-\varepsilon}^{\hat{\lambda}+\varepsilon}\right) \tag{B.62}$$

which, in conjunction with equation (B.61), implies that

$$\lim_{\lambda\to 1^-}\breve{r}_{\mathrm{y}}(\lambda)-\sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left(\breve{r}_{\mathrm{y}}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right)=$$
$$-\sum_{i=1}^{\hat{n}}\lim_{\varepsilon\to 0^+}\left(\breve{r}_{\mathrm{y}}(\lambda)\Big|_{\hat{\lambda}_i-\varepsilon}^{\hat{\lambda}_i+\varepsilon}\right)+\int_0^1\frac{1}{(1-\lambda)^2}\breve{r}_{\Delta x}(\lambda)d\lambda-\int_0^1\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda$$
$$\implies\lim_{\lambda\to 1^-}\breve{r}_{\mathrm{y}}(\lambda)=\int_0^1\frac{1}{(1-\lambda)^2}\breve{r}_{\Delta x}(\lambda)d\lambda-\int_0^1\breve{r}_{\hat{\mathrm{y}}}(\lambda)d\lambda. \tag{B.63}$$

□

B.6 Bounding Normalized Squared Residual Sums

From equation (B.2), for $\varepsilon\in(0,1)$,

$$(1-\lambda)\breve{r}_{\mathrm{y}}(\lambda)\le(1-\lambda)\breve{r}_{\mathrm{y}}(\lambda)+(1-\lambda)^2\breve{r}_{\hat{\mathrm{y}}}(\lambda)+\lambda\breve{r}_{\Delta x}(\lambda)\le$$
$$(1-\lambda)\breve{r}_{\mathrm{y}}(\varepsilon)+(1-\lambda)^2\breve{r}_{\hat{\mathrm{y}}}(\varepsilon)+\lambda\breve{r}_{\Delta x}(\varepsilon), \tag{B.64a}$$
$$\lambda\breve{r}_{\Delta x}(\lambda)\le(1-\lambda)\breve{r}_{\mathrm{y}}(\lambda)+(1-\lambda)^2\breve{r}_{\hat{\mathrm{y}}}(\lambda)+\lambda\breve{r}_{\Delta x}(\lambda)\le$$
$$(1-\lambda)\breve{r}_{\mathrm{y}}(1-\varepsilon)+(1-\lambda)^2\breve{r}_{\hat{\mathrm{y}}}(1-\varepsilon)+\lambda\breve{r}_{\Delta x}(1-\varepsilon), \tag{B.64b}$$

which implies, in conjunction with Equations (B.6) and (B.9), that

$$(1-\lambda)\breve{r}_{\mathrm{y}}(\lambda) \leq$$

$$\lim_{\varepsilon\to 0^+}\left((1-\lambda)\breve{r}_{\mathrm{y}}(\varepsilon)+(1-\lambda)^2\breve{r}_{\hat{\mathrm{y}}}(\varepsilon)+\lambda\breve{r}_{\Delta x}(\varepsilon)\right)=\lambda\lim_{\varepsilon\to 0^+}\breve{r}_{\Delta x}(\varepsilon), \tag{B.65a}$$

$$\lambda\breve{r}_{\Delta x}(\lambda)\leq\lim_{\varepsilon\to 0^+}\left((1-\lambda)\breve{r}_{\mathrm{y}}(1-\varepsilon)+(1-\lambda)^2\breve{r}_{\hat{\mathrm{y}}}(1-\varepsilon)+\lambda\breve{r}_{\Delta x}(1-\varepsilon)\right)=$$

$$(1-\lambda)\lim_{\varepsilon\to 0^+}\breve{r}_{\mathrm{y}}(1-\varepsilon)+(1-\lambda)^2\lim_{\varepsilon\to 0^+}\breve{r}_{\hat{\mathrm{y}}}(1-\varepsilon). \tag{B.65b}$$

Thus,

$$\breve{r}_{\mathrm{y}}(\lambda)\leq\frac{\lambda}{(1-\lambda)}\lim_{\varepsilon\to 0^+}\breve{r}_{\Delta x}(\varepsilon), \tag{B.66a}$$

$$\breve{r}_{\Delta x}(\lambda)\leq\frac{(1-\lambda)}{\lambda}\Big(\lim_{\varepsilon\to 0^+}\breve{r}_{\mathrm{y}}(1-\varepsilon)+(1-\lambda)\lim_{\varepsilon\to 0^+}\breve{r}_{\hat{\mathrm{y}}}(1-\varepsilon)\Big)\leq$$

$$\frac{(1-\lambda)}{\lambda}\Big(\lim_{\varepsilon\to 0^+}\breve{r}_{\mathrm{y}}(1-\varepsilon)+\lim_{\varepsilon\to 0^+}\breve{r}_{\hat{\mathrm{y}}}(1-\varepsilon)\Big), \tag{B.66b}$$

and for reasonable models, with numerical solutions that fit data at least twice as well as the homogeneous model, where $\lim_{\varepsilon\to 0^+}\breve{r}_{\mathrm{y}}(1-\varepsilon)\leq 1/2$ and $\lim_{\varepsilon\to 0^+}\breve{r}_{\hat{\mathrm{y}}}(1-\varepsilon)\leq 1/2$, and with differential equation values that fit discretized data at least as well as the homogeneous model, where $\lim_{\varepsilon\to 0^+}\breve{r}_{\Delta x}(\varepsilon)\leq 1$,

$$\breve{r}_{\mathrm{y}}(\lambda)\leq\frac{\lambda}{(1-\lambda)}, \tag{B.67a}$$

$$\breve{r}_{\Delta x}(\lambda)\leq\frac{(1-\lambda)}{\lambda}. \tag{B.67b}$$

Thus, for any data and any reasonable model of the data, to ensure that $\breve{r}_{\mathrm{y}}(\lambda)$ does not exceed some tolerance, $\bar{\varepsilon}$, λ should be chosen small enough, such that

$$\lambda\leq\frac{\bar{\varepsilon}}{1+\bar{\varepsilon}}, \tag{B.68}$$

and to ensure $\breve{r}_{\Delta x}(\lambda)$ does not exceed some tolerance, $\bar{\varepsilon}$, λ should be chosen large enough, such that

$$\lambda\geq\frac{1}{1+\bar{\varepsilon}}. \tag{B.69}$$

Appendix C

Overlapping-Niche Descent

Overlapping-niche descent, a genetic algorithm directed by gradient-based descent, synergistically minimizes $r(\mathbf{p}, \mathbf{x}; \lambda)$ over a broad range of λ values. Here, I describe overlapping-niche descent in full.

C.1 Defining Overlapping-Niche Descent

In overlapping-niche descent, I define an environment that contains n_{niche} niches, each defined by a unique value of λ, $\lambda_1 < \lambda_2 < \cdots < \lambda_{n_{\text{niche}}} \in (0, 1)$. Individuals, points of parameters and state values, inhabit the environment. Initially, in the first generation, I randomly generate parameters and state values in all individuals. In generation g, the i^{th} niche sustains $n_{g,i}$ individuals, $(\mathbf{p}_{g,i,j}, \mathbf{x}_{g,i,j})$ for $j = 1, 2, \ldots, n_{g,i}$. Starting from $(\mathbf{p}_{g,i,j}, \mathbf{x}_{g,i,j})$, I locally minimize $r(\mathbf{p}, \mathbf{x}; \lambda_i)$ to determine $(\breve{\mathbf{p}}_{g,i,j}, \breve{\mathbf{x}}_{g,i,j})$, for $i = 1, 2, \ldots, n_{\text{niche}}$ and $j = 1, 2, \ldots, n_{g,i}$, using the descent method described in Section C.2.

I decompose the number of sustained individuals in each niche, $n_{g,i}$, into the number of sustained parents, $\hat{n}_i$, which remains fixed over generations, and the number of sustained offspring, $\breve{n}_{g,i}$, which may change over generations. From $\big\{\{\breve{\mathbf{p}}_{g,i,j}, \breve{\mathbf{x}}_{g,i,j}\} \colon i \in \{1, 2, \ldots, n_{\text{niche}}\}, j \in \{1, 2, \ldots, n_{g,i}\}\big\}$, I select the $\hat{n}_i$ individuals with the $\hat{n}_i$ least values of $r(\mathbf{p}, \mathbf{x}; \lambda_i)$ to occupy the $\hat{n}_i$ parent spaces of the i^{th} niche in the $g + 1^{\text{th}}$ generation, for $i = 1, 2, \ldots, n_{\text{niche}}$; I allow individuals to occupy parent spaces in multiple niches, to permit cross-niche minimization, but do not allow individuals to occupy multiple parent spaces in the same niche, to maintain diversity in the parameter-state value search space. I enumerated individuals occupying the $\hat{n}_i$ parent spaces of the i^{th} niche from 1 to $\hat{n}_i$, such that $r(\mathbf{p}_{g+1,i,1}, \mathbf{x}_{g+1,i,1}; \lambda_i) \leq r(\mathbf{p}_{g+1,i,2}, \mathbf{x}_{g+1,i,2}; \lambda_i) \leq \cdots \leq r(\mathbf{p}_{g+1,i,\hat{n}_i}, \mathbf{x}_{g+1,i,\hat{n}_i}; \lambda_i)$, for $i = 1, 2, \ldots, n_{\text{niche}}$. Thus, for the j^{th} parent space of the i^{th} niche, I calculate the relative change in $r(\mathbf{p}, \mathbf{x}; \lambda)$ over the $g + 1^{\text{th}}$ generation:

$$\Delta r_{g+1,i,j} = \frac{r(\mathbf{p}_{g,i,j}, \mathbf{x}_{g,i,j}; \lambda_i) - r(\mathbf{p}_{g+1,i,j}, \mathbf{x}_{g+1,i,j}; \lambda_i)}{r(\mathbf{p}_{g,i,j}, \mathbf{x}_{g,i,j}; \lambda_i)} \in [0, 1]. \tag{C.1}$$

I decompose the number of sustained offspring in each niche, $\breve{n}_{g,i}$, into the number of high momentum offspring, $\breve{n}^m_{g,i}$, the number of cross-niche offspring, $\breve{n}^c_{g,i}$, the number of sexual offspring, $\breve{n}^s_{g,i}$, and the number of random offspring, $\breve{n}^r_{g,i}$. I generate high momentum offspring to accelerate convergence rates in the descent method for individuals that occupy a parent space in some niche. The individual that occupies the j^{th} parent space in the i^{th} niche also occupies

the j^{th} high momentum offspring space in the i^{th} niche. Thus, $\check{n}^m_{g,i} = \hat{n}_i$ for $i = 1, 2, \ldots, n_{\text{niche}}$. I describe details of high momentum offspring in Section C.2.2.

An individual occupying a parent space in one niche may be close to a global minimum in another niche. I generate cross-niche offspring to synergistically minimize $r(\mathbf{p}, \mathbf{x}; \lambda)$ across niches. From the set of individuals not selected from the k^{th} niche and occupying a parent space not in the k^{th} niche, $\big\{\{\check{\mathbf{p}}_{g,i,j}, \check{\mathbf{x}}_{g,i,j}\}\colon i \in \{1, 2, \ldots, n_{\text{niche}}\}\backslash\{k\}, j \in \{1, 2, \ldots, n_{g,i}\}\big\}\bigcap\big\{\{\mathbf{p}_{g+1,i,j}, \mathbf{x}_{g+1,i,j}\}\colon i \in \{1, 2, \ldots, n_{\text{niche}}\}\backslash\{k\}, j \in \{1, 2, \ldots, \hat{n}_i\}\big\}$, I randomly select an individual to occupy the l^{th} cross-niche offspring space in the k^{th} niche, with the probability of selection proportional to an individual's fitness within the k^{th} niche, $1/r(\mathbf{p}_{g+1,i,j}, \mathbf{x}_{g+1,i,j}; \lambda_k)^{q_{\text{fit}}}$, where the parameter q_{fit} dictates the strength of selection.

I generate sexual offspring to search for global minima beyond functional basins surrounding local minima. From the set of individuals occupying a parent space in some niche, $\big\{\{\mathbf{p}_{g+1,i,j}, \mathbf{x}_{g+1,i,j}\}\colon i \in \{1, 2, \ldots, n_{\text{niche}}\}, j \in \{1, 2, \ldots, \hat{n}_i\}\big\}$, I randomly select two, not necessarily distinct, sexual parents to produce the l^{th} sexual offspring in the k^{th} niche, with the probability of selection proportional to an individual's fitness within the k^{th} niche, $1/r(\mathbf{p}_{g+1,i,j}, \mathbf{x}_{g+1,i,j}; \lambda_k)^{q_{\text{fit}}}$. To produce the l^{th} sexual offspring in the k^{th} niche, I randomly combine and perturb parameters and state values from both sexual parents, a process that is isomorphic to chromosomal crossover and mutation in biological sexual reproduction. I treat each parameter and all states like separate chromosomes, independent structural units of information, and implement crossover and mutation in each chromosomal-like unit. For each parameter, a sexual offspring inherits a parameter value, with random perturbation, from one of its sexual parents, which I choose randomly with equal probability. For each sexual offspring, I choose a crossover location randomly from two to the number of elements in the numerical discretization, $\mathcal{I}_\Delta$, with equal probability. A sexual offspring inherits all state values, with random perturbations, at grid points preceding its crossover location from one its sexual parents, which I choose randomly with equal probability, and inherits all remaining state values, with random perturbations, from its other sexual parent. For effective and efficient global minimization, searches for global minima around local minima should start broad, and should narrow as individuals approach global minima. Naturally, as individuals approach global minima, $r(\mathbf{p}_{g+1,i,j}, \mathbf{x}_{g+1,i,j}; \lambda_k)^{q_{\text{fit}}}$ decreases significantly with decreasing $|\lambda_i - \lambda_k|$, increasing the likelihood of similarity in sexual parent pairings. Concomitantly, as individuals approach global minima, random perturbations in inherited parameters and state values should decrease. I measure convergence in the j^{th} parent space of the i^{th} niche by $\Delta r_{g+1,i,j}$ (C.1). As $\Delta r_{g+1,i,j}$ decreases, I decrease random perturbations in parameters and state values inherited from individual $(\mathbf{p}_{g+1,i,j}, \mathbf{x}_{g+1,i,j})$.

Parameters and state values in individuals likely do not initially cover the parameter-state value domain, and progressively cover less of the domain with successive generations of overlapping-niche descent. I generate random offspring, with random parameters and state values, to continually probe diverse regions of the parameter-state value domain for global minima.

Ultimately, I terminate overlapping-niche descent when the relative change in $r(\mathbf{p}, \mathbf{x}; \lambda)$ over the $g + 1^{\text{th}}$ generation is less than some tolerance, $\varepsilon_{\Delta r}$, in all parent spaces of all niches,

$$\Delta r_{g+1,i,j} < \varepsilon_{\Delta r}, \forall i \in \{1, 2, \ldots, n_{\text{niche}}\} \text{ and } \forall j \in \{1, 2, \ldots, \hat{n}_i\}. \tag{C.2}$$

C.2 Defining Descent

$r(\mathbf{p}, \mathbf{x}; \lambda)$ is a high dimensional system for models of complex behavior. The likelihood of randomly selecting the parameters and state values of a local minimum of $r(\mathbf{p}, \mathbf{x}; \lambda)$ decreases with increasing dimensionality, particularly with limited parameter and state value estimates. Thus, I use directed minimization to find local minimums of $r(\mathbf{p}, \mathbf{x}; \lambda)$.

Calculating the gradient of $r(\mathbf{p}, \mathbf{x}; \lambda)$ requires calculating $n(\mathbf{p}) + n(\mathbf{x})$ partial derivatives, where $n(\mathbf{v})$ denotes the number of elements in vector $\mathbf{v}$. Thus, an iteration of minimization oriented by the gradient of $r(\mathbf{p}, \mathbf{x}; \lambda)$ is relatively computationally efficient. Alternatively, $r(\mathbf{p}, \mathbf{x}; \lambda)$ is minimizable using Newton's method or a variant of Newton's method, such as the Gauss-Newton method. However, beyond calculating partial derivatives, an iteration of Newton's method or a variant of Newton's method requires solving a $(n(\mathbf{p}) + n(\mathbf{x})) \times (n(\mathbf{p}) + n(\mathbf{x}))$ linear system of equations, which can require from $O(n(\mathbf{p}) + n(\mathbf{x}))^2$ operations to solve using an iterative method, such as the generalized minimal residual method, and up to $O(n(\mathbf{p}) + n(\mathbf{x}))^3$ operations to solve using Gaussian elimination. Thus, the computational time of an iteration of $r(\mathbf{p}, \mathbf{x}; \lambda)$ minimization using Newton's method or a variant of Newton's method increases superlinearly in the number of parameters and state values. Whereas, the computational time of an iteration of $r(\mathbf{p}, \mathbf{x}; \lambda)$ minimization oriented by its gradient increases linearly in the number of parameters and state values. Thus, in minimizing $r(\mathbf{p}, \mathbf{x}; \lambda)$, to maintain computational feasibility for complex models with a large number of parameters and state values, I implement minimization oriented by the gradient of $r(\mathbf{p}, \mathbf{x}; \lambda)$ rather than using Newton's method or a variant of Newton's method.

C.2.1 Descent Scaling

The most straightforward direction for $r(\mathbf{p}, \mathbf{x}; \lambda)$ minimization is down the gradient, the direction of the negative gradient, the direction in which $r(\mathbf{p}, \mathbf{x}; \lambda)$ decreases most rapidly. However, individual parameters affect $r(\mathbf{p}, \mathbf{x}; \lambda)$ more extensively than individual state values, as a single parameter spans many differences between differential equation values and finite difference values or many differences between observable state values and data values, while a single state value spans only several differences between differential equation values and finite difference values, several differences between observable state values and data values, and several finite difference normalization values. Also, as in models with both linear and nonlinear terms, parameters vary in the degree to which they affect differences between differential equation values and finite difference values. Thus, partial derivatives of $r(\mathbf{p}, \mathbf{x}; \lambda)$ vary dramatically in

scale, in ways that do not inform distances to local minimum of $r(\mathbf{p}, \mathbf{x}; \lambda)$. So, a move down the gradient of $r(\mathbf{p}, \mathbf{x}; \lambda)$, to a lower value of $r(\mathbf{p}, \mathbf{x}; \lambda)$, is a leading order move down directions of affect-dominating variables, with lower order moves down the directions of other variables, requiring many iterations for significant changes in affect-nondominating variables. I seek a more efficient $r(\mathbf{p}, \mathbf{x}; \lambda)$ minimizing direction, one in which directional movement relates to distance from a local minimum.

For the vector $\mathbf{v} = (\mathbf{p}, \mathbf{x})$, with $n_v = n(\mathbf{p}) + n(\mathbf{x})$ elements, the directional derivative of $r(\mathbf{v}; \lambda)$, in the direction of the Hadamard product between some gradient scaling vector $\mathbf{s} = (s_1, s_2, \ldots, s_{n_v})$ and the negative gradient of $r(\mathbf{v}; \lambda)$, $-\mathbf{s} \circ \nabla r(\mathbf{v}; \lambda) = -\big(s_1 \cdot \partial_{v_1} r(\mathbf{v}; \lambda)$, $s_2 \cdot \partial_{v_2} r(\mathbf{v}; \lambda)$, $\ldots$, $s_{n_v} \cdot \partial_{v_{n_v}} r(\mathbf{v}; \lambda)\big)$, is given by

$$-\sum_{i=1}^{n_v} s_i \left(\frac{\partial r(\mathbf{v}; \lambda)}{\partial v_i} \right)^2 .$$

Thus, if $\mathbf{s}$ is a vector with positive elements, then $r(\mathbf{v}; \lambda)$ decreases in the direction of $-\mathbf{s} \circ \nabla r(\mathbf{v}; \lambda)$, at all points where $\nabla r(\mathbf{v}; \lambda) \neq \mathbf{0}$. As such, positive elements of $\mathbf{s}$ can be chosen to orient a more efficient minimization direction than down the gradient of $r(\mathbf{v}; \lambda)$. Ideally, elements of $\mathbf{s}$ should be chosen such that $\left| s_i \cdot \partial_{v_i} r(\mathbf{v}; \lambda) \right|$ is the distance along v_i to the nearest local minimum of $r(\mathbf{v}; \lambda)$. Then, a single step in the direction of $-\mathbf{s} \circ \nabla r(\mathbf{v}; \lambda)$ would lead directly to the nearest local minimum of $r(\mathbf{v}; \lambda)$. In practice, such distances are unknown, but can be approximated. A local minimum of a function occurs at a point where all partial derivatives of the function are zero. For improved efficiency over gradient descent, the computation burden of approximating distances to a local minimum of $r(\mathbf{v}; \lambda)$ should not exceed the amount of computation required for a commensurate number of gradient descent iterations. Rather than a computationally expensive search for the location where all partial derivatives of $r(\mathbf{v}; \lambda)$ are zero, I more coarsely and computationally efficiently approximate the distance along v_i to the point where all partial derivatives of $r(\mathbf{v}; \lambda)$ are zero as the distance to the zero of the linearization of $\partial_{v_i} r(\mathbf{v}; \lambda)$, with all variables constant but v_i, equivalent to the one-dimensional Newton-method approximate distance to a zero of $\partial_{v_i} r(\mathbf{v}; \lambda)$ along v_i:

$$\tilde{d}_i = \left| \frac{\partial r(\mathbf{v}; \lambda)}{\partial v_i} \right| \left| \frac{\partial^2 r(\mathbf{v}; \lambda)}{\partial v_i^2} \right|^{-1} . \tag{C.3}$$

Near a local minimum of $r(\mathbf{v}; \lambda)$, $\tilde{d}_i$ fairly accurately estimates the distance to the local minimum along v_i, as $r(\mathbf{v}; \lambda)$ is continuous. Far from a local minimum of $r(\mathbf{v}; \lambda)$, $\tilde{d}_i$ does not accurately estimate the distance to the local minimum along v_i, but does inform the scale of the distance to the local minimum along v_i better than the value of $\partial_{v_i} r(\mathbf{v}; \lambda)$ alone. Near a critical point of $r(\mathbf{v}; \lambda)$, if $\partial_{v_i v_i} r(\mathbf{v}; \lambda) > 0$, then $\tilde{d}_i$ is an approximate distance to a local minimum of $r(\mathbf{v}; \lambda)$ along v_i. For i such that $\partial_{v_i v_i} r(\mathbf{v}; \lambda) > 0$, I calculate the i^{th} element of the gradient scaling

vector $\mathbf{s}$, s_i, by equating $\left|s_i \cdot \partial_{v_i} r(\mathbf{v};\lambda)\right|$ and $\tilde{d}_i$. Thus,

$$s_i = \frac{\partial^2 r(\mathbf{v};\lambda)}{\partial v_i^2}^{-1}. \tag{C.4}$$

Near a critical point of $r(\mathbf{v};\lambda)$, if $\partial_{v_i v_i} r(\mathbf{v};\lambda) < 0$, then $\tilde{d}_i$ is an approximate distance to a local maximum of $r(\mathbf{v};\lambda)$ along v_i. For i such that $\partial_{v_i v_i} r(\mathbf{v};\lambda) \leq 0$, I choose s_i to be the search scale from the previous iteration of descent, rather than calculating s_i by equating $\left|s_i \cdot \partial_{v_i} r(\mathbf{v};\lambda)\right|$ and $\tilde{d}_i$, which does not reflect the scale of the distance to a local minimum of $r(\mathbf{v};\lambda)$ along v_i and could impede descent. For the first iteration of descent, if $\partial_{v_i v_i} r(\mathbf{v};\lambda) \leq 0$, I choose the value of s_i to be $s_{i,0}$, some very small value or zero. It may be beneficial to assign a nonzero value to $s_{i,0}$, under certain restrictions on variables, as discussed in Section C.2.3. If $s_i = 0$ and $\partial_{v_i} r(\mathbf{v};\lambda) \neq 0$, then near a local minimum of $r(\mathbf{v};\lambda)$, descent in other variables should lead to a new set of variable values where $\partial_{v_i v_i} r(\mathbf{v};\lambda) > 0$, and thus $s_i > 0$. If descent is not possible or a descent sequence does not lead to a set of variable values where $\partial_{v_i v_i} r(\mathbf{v};\lambda) > 0$, then, near a local minimum of $r(\mathbf{v};\lambda)$, some crossover and mutation event should eventually produce a set of variable values where $\partial_{v_i v_i} r(\mathbf{v};\lambda) > 0$, allowing unperturbed local minimization to continue. Collectively, denoting $\mathbf{v}_j$ and $\mathbf{s}_j$ as the vectors $\mathbf{v}$ and $\mathbf{s}$ in the j^{th} iteration of descent, with i^{th} elements $v_{i,j}$ and $s_{i,j}$, I define $s_{i,j}$, for $j \geq 1$ and $i \in \{1, 2, \ldots, n_v\}$, such that

$$s_{i,j} = \begin{cases} \dfrac{\partial^2 r(\mathbf{v}_j;\lambda)}{\partial v_{i,j}^2}^{-1} & \text{if } \dfrac{\partial^2 r(\mathbf{v}_j;\lambda)}{\partial v_{i,j}^2} > 0 \\ s_{i,j-1} & \text{if } \dfrac{\partial^2 r(\mathbf{v}_j;\lambda)}{\partial v_{i,j}^2} \leq 0. \end{cases} \tag{C.5}$$

Thus, I orient $r(\mathbf{v}_j;\lambda)$ descent in the direction of

$$\mathbf{v}_j^{\downarrow} = -\mathbf{s}_j \circ \nabla r(\mathbf{v}_j;\lambda) = -\left(s_{1,j}\frac{\partial r(\mathbf{v}_j;\lambda)}{\partial v_{1,j}}, s_{2,j}\frac{\partial r(\mathbf{v}_j;\lambda)}{\partial v_{2,j}}, \ldots, s_{n_v,j}\frac{\partial r(\mathbf{v}_j;\lambda)}{\partial v_{n_v,j}}\right). \tag{C.6}$$

Because of rough distance approximations to a local minimum of $r(\mathbf{v};\lambda)$, an extra scaling parameter of $\mathbf{v}_j^{\downarrow}$, σ_j, is required to fine-tune descent. Thus, $r(\mathbf{v};\lambda)$ descent occurs by moving away from the point $\mathbf{v}_j$, in the direction of $\sigma_j \mathbf{v}_j^{\downarrow}$, to the new point $\mathbf{v}_j + \sigma_j \mathbf{v}_j^{\downarrow}$. σ_j must be chosen such that $r(\mathbf{v}_j + \sigma_j \mathbf{v}_j^{\downarrow};\lambda) < r(\mathbf{v}_j;\lambda)$, and should be chosen large enough to avoid an excess number of descent iterations. To choose σ_j, I begin with a value of $\sigma_j = 1$. If $r(\mathbf{v}_j + 2\mathbf{v}_j^{\downarrow};\lambda) < r(\mathbf{v}_j + \mathbf{v}_j^{\downarrow};\lambda) < r(\mathbf{v}_j;\lambda)$, I expand σ_j, continuing to double σ_j until $r(\mathbf{v}_j + 2\sigma_j \mathbf{v}_j^{\downarrow};\lambda) \geq r(\mathbf{v}_j + \sigma_j \mathbf{v}_j^{\downarrow};\lambda)$. If, however, $r(\mathbf{v}_j + \mathbf{v}_j^{\downarrow};\lambda) \geq r(\mathbf{v}_j;\lambda)$ or $r(\mathbf{v}_j + 2\mathbf{v}_j^{\downarrow};\lambda) \geq r(\mathbf{v}_j + \mathbf{v}_j^{\downarrow};\lambda)$, I contract σ_j, continuing to half σ_j until $r(\mathbf{v}_j + 2^{-1}\sigma_j \mathbf{v}_j^{\downarrow};\lambda) \geq r(\mathbf{v}_j + \sigma_j \mathbf{v}_j^{\downarrow};\lambda) < r(\mathbf{v}_j;\lambda)$ or until σ_j contracts below some specified tolerance ε_σ.

Descent maps $v_{i,j}$ to $v_{i,j}^d$:

$$v_{i,j}^d = v_{i,j} - \sigma_j s_{i,j} \frac{\partial r(\mathbf{v}_j; \lambda)}{\partial v_{i,j}}. \tag{C.7}$$

Under the variable scaling

$$\hat{v}_{i,j} = s_{i,j}^{-1/2} v_{i,j}, \forall i \in \{1, 2, \ldots, n_v\}, \tag{C.8}$$

Equation (C.7) becomes

$$v_{i,j}^d = s_{i,j}^{1/2} \hat{v}_{i,j} - \sigma_j s_{i,j} s_{i,j}^{-1/2} \frac{\partial r(\hat{\mathbf{v}}_j; \lambda)}{\partial \hat{v}_{i,j}} = s_{i,j}^{1/2} \left(\hat{v}_{i,j} - \sigma_j \frac{\partial r(\hat{\mathbf{v}}_j; \lambda)}{\partial \hat{v}_{i,j}} \right), \tag{C.9}$$

as

$$\frac{\partial r(\mathbf{v}_j; \lambda)}{\partial v_{i,j}} = \frac{\partial r(\hat{\mathbf{v}}_j; \lambda)}{\partial \hat{v}_{i,j}} \frac{d\hat{v}_{i,j}}{dv_{i,j}} = s_{i,j}^{-1/2} \frac{\partial r(\hat{\mathbf{v}}_j; \lambda)}{\partial \hat{v}_{i,j}}. \tag{C.10}$$

Thus, a step of descent is equivalent to a step of gradient descent under the variable scaling in equation (C.8).

C.2.2 Descent Acceleration

With very little extra computational burden, Nesterov's method significantly increases the convergence rate of gradient descent in functional minimization [50]. In Nesterov's method, movement from the point of values in the j^{th} iteration, along the direction of the change in values over the j^{th} iteration, generates an intermediary point of values. Then, movement from the intermediary point of values, down the gradient of the functional, generates the point of values in the $j + 1^{\text{th}}$ iteration. Although Nesterov's method converges to a local minimum of a functional, it is not a descent method, as the functional value may increase during some iterations. Restarting Nesterov's method when functional values would increase during an iteration ensures descent during every iteration and accelerates the method's rate of convergence [51]. To minimize $r(\mathbf{v}; \lambda)$, I apply Nesterov's method with increasing-functional restart from

[51], with variable scaling, strict descent, and termination-tolerance restart modifications:

$$\theta_j = \begin{cases} 1 & \text{if } j = 1 \text{ or } \delta_j = 0 \text{ or } \tau_j = 1 \\ \theta_{j-1}\left(-\theta_{j-1} + (\theta_{j-1}^2 + 4)^{1/2}\right)/2 & \text{otherwise,} \end{cases} \tag{C.11a}$$

$$\beta_j = \begin{cases} 0 & \text{if } j = 1 \text{ or } \delta_j = 0 \text{ or } \tau_j = 1 \\ \theta_{j-1}(1 - \theta_{j-1})/(\theta_{j-1}^2 + \theta_j) & \text{otherwise,} \end{cases} \tag{C.11b}$$

$$\mathbf{v}_j = \begin{cases} \mathbf{u}_j & \text{if } j = 1 \text{ or } \delta_j = 0 \text{ or } \tau_j = 1 \\ \mathbf{u}_j + \beta_j(\mathbf{u}_j - \mathbf{u}_{j-1}) & \text{otherwise,} \end{cases} \tag{C.11c}$$

$$\mathbf{u}_{j+1} = \begin{cases} \mathbf{v}_j + \sigma_j \mathbf{v}_j^{\downarrow} & \text{if } r(\mathbf{v}_j + \sigma_j \mathbf{v}_j^{\downarrow}; \lambda) < r(\mathbf{u}_j; \lambda) \\ \mathbf{u}_j & \text{otherwise,} \end{cases} \tag{C.11d}$$

$$\delta_{j+1} = \begin{cases} 1 & \text{if } r(\mathbf{u}_{j+1}; \lambda) < r(\mathbf{u}_j; \lambda) \\ 0 & \text{otherwise,} \end{cases} \tag{C.11e}$$

$$\tau_{j+1} = \begin{cases} 1 & \text{if } \sigma_j < \varepsilon_\sigma \text{ or } \big(r(\mathbf{u}_j; \lambda) - r(\mathbf{u}_{j+1}; \lambda)\big)/r(\mathbf{u}_j; \lambda) < \varepsilon_r \\ 0 & \text{otherwise,} \end{cases} \tag{C.11f}$$

where $\mathbf{u}_j$ is the vector of parameter and states values in iteration $j \geq 1$, θ_j tempers the iterational increase from 0 to 1 in β_j, the proportion along the iterational change in $\mathbf{u}_j$, which generates the intermediary point $\mathbf{v}_j$; δ_j is the strict descent indicator, and τ_j is the termination tolerance indicator. Accelerated descent begins with scaled gradient descent, and restarts if either strict descent does not occur, $\delta_j = 0$, or a termination tolerance is met, $\tau_j = 1$. A termination tolerance is met if either the fine-tuning scaling parameter, σ_j, contracts below the specified tolerance, ε_σ, or the iterational relative change in $r(\mathbf{u}_j; \lambda)$ falls below the specified tolerance, ε_r. Ultimately, accelerated descent terminates in iteration j if a termination tolerance is met after restart, $\beta_{j-1} = 0$ and $\tau_j = 1$, during an iteration of scaled gradient descent. Alternatively, accelerated descent terminates in iteration j if the number of strict descent iterations reaches some specified maximum number of strict descent iterations, $n_{\max}$, which occurs when $\sum_{i=2}^{j} \delta_i = n_{\max}$.

To maintain momentum in convergence over generations of accelerated descent, an individual that occupies a parent space in a niche begins accelerated descent with θ_{j-1} and $\mathbf{u}_{j-1}$ from its last iteration of accelerated descent in the previous generation, rather than beginning accelerated descent with a restart. To overcome stagnating convergence, for an individual that occupies a parent space in a niche, I extend momentum from its last iteration of accelerated descent to its last generation of accelerated descent, in a high momentum offspring. Thus, the individual that occupies the l^{th} high momentum offspring space in the k^{th} niche of the g^{th} generation begins accelerated descent with θ_{j-1} from its last iteration of accelerated descent in the $g-1^{\text{th}}$ generation, and begins accelerated descent with $\mathbf{u}_{j-1} = (\mathbf{p}_{g-1,k,l}, \mathbf{x}_{g-1,k,l})$, the parameters and state values of the individual that occupies the l^{th} parent space in the k^{th} niche of the $g-1^{\text{th}}$ generation.

C.2.3 Descent on Restricted Domains

When parameters and states are defined on a restricted domain, accelerated descent trajectories must be confined to the restricted domain. Generally, accelerated descent would restart when $\mathbf{u}_j$ or $\mathbf{v}_j$ would leave the restricted domain, and the gradient scaling vector, $\mathbf{s}_j$, with scaling parameter, σ_j, would be chosen to ensure that $\mathbf{u}_{j+1}$ remains within the restricted domain. Less onerously, when the restricted domain is a closed convex set, I can simply project accelerated descent trajectories onto the restricted domain.

From the Bourbaki-Cheney-Goldstein inequality, for some point $\mathbf{x}$ and a closed convex set C,

$$\begin{gathered}\langle \mathbf{x} - P_C(\mathbf{x}), P_C(\mathbf{x}) - \mathbf{y} \rangle \geq 0 \text{ for all } \mathbf{y} \in C, \text{ where} \\ P_C(\mathbf{u}) = \arg\min\{\|\mathbf{v} - \mathbf{u}\| : \mathbf{v} \in C\}\end{gathered} \tag{C.12}$$

and $\langle \cdot, \cdot \rangle$ denotes the standard inner product. Thus, for a functional $f(\mathbf{u})$, $\mathbf{x}_0 \in C$, and $\mathbf{x} = \mathbf{x}_0 - \sigma \nabla f(\mathbf{x}_0)$ for some $\sigma > 0$, if $P_C(\mathbf{x}) \neq \mathbf{x}_0$, then

$$\begin{gathered}\langle \mathbf{x} - P_C(\mathbf{x}), P_C(\mathbf{x}) - \mathbf{x}_0 \rangle \geq 0 \iff \\ \langle \mathbf{x}_0 - \sigma \nabla f(\mathbf{x}_0) - P_C(\mathbf{x}), P_C(\mathbf{x}) - \mathbf{x}_0 \rangle \geq 0 \iff \\ \langle \sigma \nabla f(\mathbf{x}_0) + P_C(\mathbf{x}) - \mathbf{x}_0, P_C(\mathbf{x}) - \mathbf{x}_0 \rangle \leq 0 \iff \\ \langle \sigma \nabla f(\mathbf{x}_0), P_C(\mathbf{x}) - \mathbf{x}_0 \rangle + \langle P_C(\mathbf{x}) - \mathbf{x}_0, P_C(\mathbf{x}) - \mathbf{x}_0 \rangle \leq 0 \implies \\ \langle \nabla f(\mathbf{x}_0), P_C(\mathbf{x}) - \mathbf{x}_0 \rangle \leq -\sigma^{-1} \langle P_C(\mathbf{x}) - \mathbf{x}_0, P_C(\mathbf{x}) - \mathbf{x}_0 \rangle < 0 \implies \\ \big\langle \nabla f(\mathbf{x}_0), P_C\big(\mathbf{x}_0 - \sigma \nabla f(\mathbf{x}_0)\big) - \mathbf{x}_0 \big\rangle < 0,\end{gathered} \tag{C.13}$$

and, if $P_C(\mathbf{x}) = \mathbf{x}_0$, then

$$\begin{gathered}\langle \mathbf{x} - \mathbf{x}_0, \mathbf{x}_0 - \mathbf{y} \rangle \geq 0 \ \forall \mathbf{y} \in C \iff \\ \langle \mathbf{x}_0 - \sigma \nabla f(\mathbf{x}_0) - \mathbf{x}_0, \mathbf{x}_0 - \mathbf{y} \rangle \geq 0 \ \forall \mathbf{y} \in C \iff \\ \langle -\sigma \nabla f(\mathbf{x}_0), \mathbf{x}_0 - \mathbf{y} \rangle \geq 0 \ \forall \mathbf{y} \in C \iff \\ \langle \nabla f(\mathbf{x}_0), \mathbf{y} - \mathbf{x}_0 \rangle \geq 0 \ \forall \mathbf{y} \in C.\end{gathered} \tag{C.14}$$

Inequality (C.13) implies that the non-invariant projection of a gradient descent trajectory onto a closed convex set is a strictly decreasing direction of a functional. Inequality (C.14) implies that a functional does not decrease locally in a closed convex set when the projection of a gradient descent trajectory onto the closed convex set is invariant. Therefore, the projection of gradient descent trajectories onto a closed convex set preserves convergence to a local minimum of a functional in the closed convex set.

The projection of a scaled descent trajectory, $\mathbf{v}_j + \sigma_j \mathbf{v}_j^{\downarrow}$, onto a closed convex set is not necessarily a decreasing direction of $r(\mathbf{v}; \lambda)$. However, from equation (C.9), scaled descent is equivalent to gradient descent under the variable transformation in equation (C.8). Therefore,

to ensure descent, when parameters and states are restricted to a closed convex set, I project scaled descent trajectories onto the restricted domain under the variable transformation in equation (C.8). I note that if $s_{i,j} = 0$ for some variable, v_i, the projection of a scaled descent trajectory onto the restricted domain may not be defined under the variable transformation in equation (C.8). In such cases, assigning $s_{i,0}$ some very small, positive value ensures that $s_{i,j} \neq 0$. Additionally, to confine accelerated descent intermediary points, $\mathbf{v}_j$, to the restricted domain, I project accelerated descent intermediary points onto the restricted domain. Accelerated descent restarts upon reaching a termination tolerance. Thus, accelerated descent ultimately terminates during an iteration of scaled gradient descent, preserving convergence to a local minimum of $r(\mathbf{v}; \lambda)$ on a restricted domain.

Often, a closed convex set C is generated from the union of n_c simpler closed convex sets $C_1, C_2, \ldots, C_{n_c}$. In such cases, I employ Dykstra's method [19] to calculate the projection of a point $\mathbf{x}$ onto $C = C_1 \cap C_2 \cap \cdots \cap C_{n_c}$:

$$\mathbf{x}_i^j = \begin{cases} \mathbf{x} & \text{if } i = 0 \\ P_{C_j}(\mathbf{x}_{i-1}^{n_c} - \mathbf{d}_{i-1}^j) & \text{if } i > 0 \text{ and } j = 1 \\ P_{C_j}(\mathbf{x}_i^{j-1} - \mathbf{d}_{i-1}^j) & \text{if } i > 0 \text{ and } j > 1, \end{cases} \tag{C.15a}$$

$$\text{where } P_{C_j}(\mathbf{u}) = \arg\min\{\|\mathbf{v} - \mathbf{u}\| : \mathbf{v} \in C_j\} \text{ and} \tag{C.15b}$$

$$\mathbf{d}_i^j = \begin{cases} \mathbf{0} & \text{if } i = 0 \\ \mathbf{x}_i^j - (\mathbf{x}_{i-1}^{n_c} - \mathbf{d}_{i-1}^j) & \text{if } i > 0 \text{ and } j = 1 \\ \mathbf{x}_i^j - (\mathbf{x}_i^{j-1} - \mathbf{d}_{i-1}^j) & \text{if } i > 0 \text{ and } j > 1, \end{cases} \tag{C.15c}$$

for $i \in \{0, 1, 2, \ldots\}$ and $j \in \{1, 2, \ldots, n_c\}$. $\|\mathbf{x}_i^j - P_C(\mathbf{x})\| \to 0$ as $i \to \infty$ and $\mathbf{x}_i^j$ converges monotonically to $P_C(\mathbf{x})$ in i and j [6]. Commonly, a linear inequality may restrict points to a closed half-space. When all C_j are closed half-spaces, the sequence $\{\mathbf{x}_i^{n_c}\}$ converges linearly to $P_C(\mathbf{x})$ [16]. I terminate Dykstra's method when $|x_{i,k}^{n_c} - x_{i-1,k}^{n_c}| < \varepsilon_c |x_{i-1,k}^{n_c}|$ or $|x_{i,k}^{n_c} - x_{i-1,k}^{n_c}| < \varepsilon_{\bar{c}}$ for all elements $x_{i,k}^{n_c}$ in $\mathbf{x}_i^{n_c}$, given some relative termination tolerance $\varepsilon_c > 0$ and some absolute termination tolerance $\varepsilon_{\bar{c}} > 0$.

C.3 Descent Prolongation

Accelerated descent terminates if the number of strict descent iterations reaches $n_{\max}$. Individuals occupying parent spaces in niches have been descending for multiple generations of overlapping-niche descent. Thus, for $n_{\max}$ not exceedingly large, an individual that occupies an offspring space in some niche may terminate accelerated descent with a higher value of $r(\mathbf{v}; \lambda)$ than an individual that occupies a parent space in the same niche, even though the individual that occupies the offspring space could ultimately converge to a lower value of $r(\mathbf{v}; \lambda)$ than the individual that occupies the parent space. In preliminary tests with randomly generated parameter and states values, I have found that following initial, rapid sublinear convergence,

accelerated descent trajectories generally converge linearly or superlinear in periods between accelerated descent restart. I prolong accelerated descent for an individual that converges linearly or superlinearly to a value of $r(\mathbf{v};\lambda)$ below the least value of $r(\mathbf{v};\lambda)$ in its niche.

Theorem 6. *For a strictly decreasing sequence $(b_k)_{k=0}^{\infty}$ converging linearly or superlinearly to b, if*

$$b_{n+2m} < \tilde{b} + \frac{(b_{n+m} - \tilde{b})^2}{b_n - \tilde{b}}, \tag{C.16}$$

then b may be less than $\tilde{b}$, and if

$$b_{n+2m} > \tilde{b} + \frac{(b_{n+m} - \tilde{b})^2}{b_n - \tilde{b}}, \tag{C.17}$$

then $b > \tilde{b}$, for $m \in \mathbb{N}$. Also, if $b > \tilde{b}$ and

$$\frac{b_{n+m} - b}{b_n - b} < \rho, \tag{C.18}$$

for some $\rho \in (0, 1/2)$, then there exists some $n \in \mathbb{N}$ such that

$$b_{n+2m} > \tilde{b} + \frac{(b_{n+m} - \tilde{b})^2}{b_n - \tilde{b}} \tag{C.19}$$

for all $n > N$.

Proof. For a strictly decreasing sequence $(a_k)_{k=0}^{\infty}$ that converges linearly to a,

$$a_{n+1} - a = \mu(a_n - a), \tag{C.20}$$

for some μ, where $0 < \mu < 1$. Thus, for $m \in \mathbb{N}$,

$$a_{n+m} - a = \mu(a_{n+m-1} - a) = \mu^2(a_{n+m-2} - a) = \cdots = \mu^m(a_n - a), \tag{C.21}$$

from which it follows that

$$a_{n+2m} - a = \mu^{2m}(a_n - a) = \left(\mu^m\right)^2(a_n - a) = \left(\frac{a_{n+m} - a}{a_n - a}\right)^2 (a_n - a) = \frac{(a_{n+m} - a)^2}{a_n - a} \iff a_{n+2m} = a + \frac{(a_{n+m} - a)^2}{a_n - a}. \tag{C.22}$$

Thus, for a strictly decreasing sequence $(b_k)_{k=0}^{\infty}$,

$$\tilde{b}_{n+2m} = \tilde{b} + \frac{(b_{n+m} - \tilde{b})^2}{b_n - \tilde{b}} \tag{C.23}$$

is the linear-convergence estimate of b_{n+2m}, with estimate, $\tilde{b}$, to the limit of the sequence, b.

$$\frac{\partial \tilde{b}_{n+2m}}{\partial \tilde{b}} = \left(\frac{b_n - b_{n+m}}{b_n - \tilde{b}}\right)^2 > 0. \tag{C.24}$$

Thus, if $(b_k)_{k=0}^{\infty}$ is converging linearly to $b < \tilde{b}$, then

$$b_{n+2m} = b + \frac{(b_{n+m} - b)^2}{b_n - b} < \tilde{b} + \frac{(b_{n+m} - \tilde{b})^2}{b_n - \tilde{b}} = \tilde{b}_{n+2m}, \tag{C.25}$$

from which it follows that if $(b_k)_{k=0}^{\infty}$ is converging superlinearly to $b < \tilde{b}$, then

$$b_{n+2m} < b + \frac{(b_{n+m} - b)^2}{b_n - b} < \tilde{b} + \frac{(b_{n+m} - \tilde{b})^2}{b_n - \tilde{b}} = \tilde{b}_{n+2m}; \tag{C.26}$$

if $(b_k)_{k=0}^{\infty}$ is converging linearly to $b > \tilde{b}$, then

$$b_{n+2m} = b + \frac{(b_{n+m} - b)^2}{b_n - b} > \tilde{b} + \frac{(b_{n+m} - \tilde{b})^2}{b_n - \tilde{b}} = \tilde{b}_{n+2m}, \tag{C.27}$$

from which it follows that if $(b_k)_{k=0}^{\infty}$ is converging sublinearly to $b > \tilde{b}$, then

$$b_{n+2m} > b + \frac{(b_{n+m} - b)^2}{b_n - b} > \tilde{b} + \frac{(b_{n+m} - \tilde{b})^2}{b_n - \tilde{b}} = \tilde{b}_{n+2m}. \tag{C.28}$$

For $(b_k)_{k=0}^{\infty}$ converging superlinearly to $b > \tilde{b}$, I determine when $b > \tilde{b}_{n+2m}$, which implies that $b_{n+2m} \geq b > \tilde{b}_{n+2m}$. For $\delta = b - \tilde{b}$ and $\varepsilon_n = b_n - b$,

$$\begin{gathered} b > \tilde{b}_{n+2m} = \tilde{b} + \frac{(b_{n+m} - \tilde{b})^2}{b_n - \tilde{b}} \iff b > b - \delta + \frac{(\varepsilon_{n+m} + \delta)^2}{\varepsilon_n + \delta} \iff \\ \varepsilon_{n+m}^2 + 2\delta\varepsilon_{n+m} - \varepsilon_n\delta < 0 \iff \varepsilon_{n+m} \in \left(-\delta - \sqrt{\delta^2 + \varepsilon_n\delta}, -\delta + \sqrt{\delta^2 + \varepsilon_n\delta}\right). \end{gathered} \tag{C.29}$$

As $\varepsilon_{n+m} \geq 0$, equation (C.29) holds, for $\varepsilon_n > 0$, if and only if

$$\varepsilon_{n+m} < -\delta + \sqrt{\delta^2 + \varepsilon_n\delta} \iff \frac{\varepsilon_{n+m}}{\varepsilon_n} < \frac{-\delta + \sqrt{\delta^2 + \varepsilon_n\delta}}{\varepsilon_n}, \tag{C.30}$$

which necessarily holds for large enough m, as $\lim_{m\to\infty}\varepsilon_{n+m}=0$.

$$\frac{\partial}{\partial\varepsilon_n}\left(\frac{-\delta+\sqrt{\delta^2+\varepsilon_n\delta}}{\varepsilon_n}\right)=\frac{-(2\delta^2+\varepsilon_n\delta)+2\delta\sqrt{\delta^2+\varepsilon_n\delta}}{2\varepsilon_n^2\sqrt{\delta^2+\varepsilon_n\delta}}, \tag{C.31a}$$

$$(2\delta^2+\varepsilon_n\delta)^2=4\delta^4+4\varepsilon_n\delta^3+\varepsilon_n^2\delta^2>4\delta^4+4\varepsilon_n\delta^3=\left(2\delta\sqrt{\delta^2+\varepsilon_n\delta}\right)^2\implies$$

$$2\delta^2+\varepsilon_n\delta>2\delta\sqrt{\delta^2+\varepsilon_n\delta}\implies$$

$$\frac{\partial}{\partial\varepsilon_n}\left(\frac{-\delta+\sqrt{\delta^2+\varepsilon_n\delta}}{\varepsilon_n}\right)<0. \tag{C.31b}$$

By L'Hôpital's rule,

$$\lim_{\varepsilon_n\to0^+}\frac{-\delta+\sqrt{\delta^2+\varepsilon_n\delta}}{\varepsilon_n}=\lim_{\varepsilon_n\to0^+}\frac{\delta}{2\sqrt{\delta^2+\varepsilon_n\delta}}=\frac{1}{2}, \tag{C.32}$$

$$\lim_{\varepsilon_n\to\infty}\frac{-\delta+\sqrt{\delta^2+\varepsilon_n\delta}}{\varepsilon_n}=\lim_{\varepsilon_n\to\infty}\frac{\delta}{2\sqrt{\delta^2+\varepsilon_n\delta}}=0, \tag{C.33}$$

and equation (C.31b),

$$0<\frac{-\delta+\sqrt{\delta^2+\varepsilon_0\delta}}{\varepsilon_0}<\cdots<\frac{-\delta+\sqrt{\delta^2+\varepsilon_n\delta}}{\varepsilon_n}<\frac{-\delta+\sqrt{\delta^2+\varepsilon_{n+1}\delta}}{\varepsilon_{n+1}}<\cdots<\frac{1}{2}. \tag{C.34}$$

Thus, for any $\rho\in(0,1/2)$ and m large enough that $\varepsilon_{n+m}/\varepsilon_n<\rho$, there exists some $N\in\mathbb{N}$ such that inequality (C.30) holds for all $n>N$, which implies that $b_{n+2m}>\tilde{b}_{n+2m}$ for all $n>N$.

Example 1. *For m such that $3\varepsilon_{n+m}<\varepsilon_n$ and N such that $n>N$ implies that $\varepsilon_n<3\delta$,*

$$\frac{\varepsilon_{n+m}}{\varepsilon_n}<\frac{1}{3}=\frac{-\delta+\sqrt{\delta^2+3\delta^2}}{3\delta}<\frac{-\delta+\sqrt{\delta^2+\varepsilon_n\delta}}{\varepsilon_n}, \tag{C.35}$$

for $n>N$, which implies that $b_{n+2m}>\tilde{b}_{n+2m}$ for $n>N$.

Collectively, if $(b_k)_{k=0}^{\infty}$ converges linearly or superlinearly to $b<\tilde{b}$, then $b_{n+2m}<\tilde{b}_{n+2m}$ for all $m,n\in\mathbb{N}$; if $(b_k)_{k=0}^{\infty}$ converges to $b>\tilde{b}$, then for sufficiently, but not exceedingly large $m\in\mathbb{N}$, $b_{n+2m}>\tilde{b}_{n+2m}$, for all $n\in\mathbb{N}$ greater than some N. Therefore, for $(b_k)_{k=0}^{\infty}$ converging linearly or superlinearly to b, if $b_{n+2m}<\tilde{b}_{n+2m}$, then b may be less than $\tilde{b}$, and if $b_{n+2m}>\tilde{b}_{n+2m}$, then b is greater than $\tilde{b}$. □

For $\breve{r}_\lambda$, the least value of $r(\mathbf{v};\lambda)$ amongst all individuals inhabiting the environment at the onset of accelerated descent, I prolong accelerated descent if the $\lim_{j\to\infty}r(\mathbf{u}_j;\lambda)$ may be less than $\breve{\sigma}\breve{r}_\lambda$, where I choose $\breve{\sigma}\in[0,1]$ to specify the stringency of prolongation. Thus, if the number of strict descent iterations has reached $n_{\max}$ and $r(\mathbf{u}_j;\lambda)>\breve{r}_\lambda$, following Theorem 6, I prolong

accelerated descent, up to $\hat{n}_{\text{pro}}$ strict descent iterations, if

$$r(\mathbf{u}_j;\lambda) < \breve{\sigma}\breve{r}_\lambda + \frac{(r(\mathbf{u}_{j-m_{\text{pro}}};\lambda) - \breve{\sigma}\breve{r}_\lambda)^2}{r(\mathbf{u}_{j-2m_{\text{pro}}};\lambda) - \breve{\sigma}\breve{r}_\lambda} \tag{C.36}$$

or if a restart occurred between iterations $j - 2m_{\text{pro}}$ and j, for some sufficiently, but not exceedingly large, chosen value of m_{pro}. If $r(\mathbf{u}_j;\lambda) \leq \breve{r}_\lambda$ and $r(\mathbf{u}_{j-1};\lambda) > \breve{r}_\lambda$, I prolong accelerated descent by $\check{n}_{\text{pro}}$ strict descent iterations, to remove bias from an uneven number of strict descent iterations in selection, when comparing individuals with values of $r(\mathbf{v};\lambda)$ that fall below $\breve{r}_\lambda$.

Appendix D

Computational Complexities

Here, I calculate and compare computational complexities of $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent and a variety of numerical-integration-based methods.

D.1 Computational Complexity of $r(\mathbf{p}, \mathbf{x}; \lambda)$ Descent

D.1.1 Formulation of $r(\mathbf{p}, \mathbf{x}; \lambda)$ for Counting

Descent is the most computationally intensive portion of overlapping-niche descent. Here, I count the computation complexity required for an iteration of $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent. I consider $r(\mathbf{p}, \mathbf{x}; \lambda)$ in the form

$$r(\mathbf{p}, \mathbf{x}; \lambda) = \frac{1-\lambda}{n_y} \sum_{j=1}^{n_y} \sum_{k=1}^{n_t} d_{\mathrm{y}_{j,k}}\big(g_{j,k}(\mathbf{p}, \mathbf{x})\big) + \frac{(1-\lambda)^2}{n_y} \sum_{j=1}^{n_y} \sum_{k \in \mathcal{I}_{\hat{\mathrm{y}}}} d_{\hat{\mathrm{y}}_{j,k}}\big(g_{j,k}(\mathbf{p}, \mathbf{x})\big) + \frac{\lambda}{n_x} \sum_{i=1}^{n_x} h_i(\mathbf{x}) \sum_{k \in \mathcal{I}_\Delta} d_{\Delta_{i,k}}\big(f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})\big), \tag{D.1}$$

where $g_{j,k}(\mathbf{p}, \mathbf{x}) = \mathrm{y}_{j,k} - g_j(\mathbf{p}, x_{1,k}, \ldots, x_{n_x,k})$, with difference measures $d_{\mathrm{y}_{j,k}}$, $d_{\hat{\mathrm{y}}_{j,k}}$, and $d_{\Delta_{i,k}}$, and auxiliary functions $h_i(\mathbf{x})$, which may contain a smoothing penalty and numerical method normalization. For example, with $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ as defined in Section 2.3, $r_{\hat{\mathrm{y}}}(\mathbf{p}, \mathbf{x})$ as defined in Section 2.6.1, and $r_{\Delta x}(\mathbf{p}, \mathbf{x})$ as defined in Section 2.4.1,

$$d_{\mathrm{y}_{j,k}}(u) = \frac{w_{j,k}}{\sum_{k=1}^{n_t} w_{j,k} \mathrm{y}_{j,k}^2} u^2, \; d_{\hat{\mathrm{y}}_{j,k}}(u) = \frac{\hat{\sigma} \hat{w}_{j,k}}{\sum_{k \in \mathcal{I}_{\hat{\mathrm{y}}}} \hat{w}_{j,k} \hat{\mathrm{y}}_{j,k}^2} u^2,$$
$$d_{\Delta_{i,k}}(u) = u^2, \; h_i(\mathbf{x}) = \frac{s_i(\mathbf{x})}{\sum_{k \in \mathcal{I}_\Delta} (\Delta x_{i,k})^2}. \tag{D.2}$$

D.1.2 Defining Quantities for Counting

Preliminarily, I define quantities for the computational complexity counting of $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent. I consider computationally simple difference measures, which require $O(1)$ operations to calculate

a function value and $O(1)$ operations to calculate a partial derivative value. Thus, calculating

$$\begin{aligned} d_{\mathrm{y}_{j,k}}(u), d_{\hat{\mathrm{y}}_{j,k}}(u), d_{\Delta_{i,k}}(u) &\text{ requires } O(1) \text{ operations,} \\ \partial d_{\mathrm{y}_{j,k}}(u), \partial d_{\hat{\mathrm{y}}_{j,k}}(u), \partial d_{\Delta_{i,k}}(u) &\text{ requires } O(1) \text{ operations,} \\ \partial^2 d_{\mathrm{y}_{j,k}}(u), \partial^2 d_{\hat{\mathrm{y}}_{j,k}}(u), \partial^2 d_{\Delta_{i,k}}(u) &\text{ requires } O(1) \text{ operations,} \end{aligned} \tag{D.3}$$

for $i \in \{1, 2, \ldots, n_x\}$, $j \in \{1, 2, \ldots, n_y\}$, and $k \in \mathcal{I}_\Delta$. I note that $d_{\mathrm{y}_{j,k}}(u)$ and $d_{\hat{\mathrm{y}}_{j,k}}(u)$ in equation (D.2) are computationally simple after initially calculating and storing $\sum_{k=1}^{n_t} w_{j,k} \mathrm{y}_{j,k}^2$ and $\sum_{k \in \mathcal{I}_{\hat{\mathrm{y}}}} \hat{w}_{j,k} \hat{\mathrm{y}}_{j,k}^2$. On average, $g_{j,k}(\mathbf{p}, \mathbf{x})$ requires n_g operations to calculate a function value, n_{g_1} operations to calculate a first order partial derivative value, and n_{g_2} operations to calculate a second order partial derivative value. Thus, I consider $g_{j,k}(\mathbf{p}, \mathbf{x})$ such that calculating

$$\begin{aligned} g_{j,k}(\mathbf{p}, \mathbf{x}) &\text{ requires } O(n_g) \text{ operations,} \\ \partial g_{j,k}(\mathbf{p}, \mathbf{x}) &\text{ requires } O(n_{g_1}) \text{ operations,} \\ \partial^2 g_{j,k}(\mathbf{p}, \mathbf{x}) &\text{ requires } O(n_{g_2}) \text{ operations,} \end{aligned} \tag{D.4}$$

for $j \in \{1, 2, \ldots, n_y\}$ and $k \in \mathcal{I}_\Delta$. On average, $f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ requires n_f operations to calculate a function value, n_{f_1} operations to calculate a first order partial derivative value, and n_{f_2} operations to calculate a second order partial derivative value. Thus, I consider $f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ such that calculating

$$\begin{aligned} f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) &\text{ requires } O(n_f) \text{ operations,} \\ \partial f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) &\text{ requires } O(n_{f_1}) \text{ operations,} \\ \partial^2 f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) &\text{ requires } O(n_{f_2}) \text{ operations,} \end{aligned} \tag{D.5}$$

for $i \in \{1, 2, \ldots, n_x\}$ and $k \in \mathcal{I}_\Delta$. Auxiliary functions, $h_i(\mathbf{x})$, modify $\sum_{k \in \mathcal{I}_\Delta} d_{\Delta_{i,k}} \circ f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ and, generally, are computationally simpler than $\sum_{k \in \mathcal{I}_\Delta} f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$. Thus, I consider $h_i(\mathbf{x})$ that are no more computationally complex than $\sum_{k \in \mathcal{I}_\Delta} f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$, with partial derivatives that are no more computationally complex than corresponding partial derivative of $\sum_{k \in \mathcal{I}_\Delta} f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$. Calculating $\sum_{k \in \mathcal{I}_\Delta} f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ requires $O(n_f n_\Delta)$ operations, where n_Δ is the number of elements in $\mathcal{I}_\Delta$. $f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ depend on $x_{l,m}$ for only a small fraction of k in $\mathcal{I}_\Delta$, at k in $\mathcal{I}_{\Delta_m} \subset \mathcal{I}_\Delta$. Thus, calculating a first order partial derivative of $\sum_{k \in \mathcal{I}_\Delta} f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ with respect to $x_{l,m}$ requires $O(n_{f_1} n_\delta)$ operations, and calculating a second order partial derivative of $\sum_{k \in \mathcal{I}_\Delta} f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ with respect to $x_{l,m}$ requires $O(n_{f_2} n_\delta)$ operations, where n_δ is the number of elements in $\mathcal{I}_{\Delta_m}$. Therefore, calculating

$$\begin{aligned} h_i(\mathbf{x}) &\text{ requires } O(\leq n_f n_\Delta) \text{ operations,} \\ \partial h_i(\mathbf{x}) &\text{ requires } O(\leq n_{f_1} n_\delta) \text{ operations,} \\ \partial^2 h_i(\mathbf{x}) &\text{ requires } O(\leq n_{f_2} n_\delta) \text{ operations,} \end{aligned} \tag{D.6}$$

for $i \in \{1, 2, \ldots, n_x\}$. On average, discretized differential equation values, $F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$, require n_F operations to calculate a function value, n_{F_1} operations to calculate a first order partial derivative value, and n_{F_2} operations to calculate a second order partial derivative value. Thus, I consider $F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ such that calculating

$$\begin{aligned} F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) &\text{ requires } O(n_F) \text{ operations,} \\ \partial F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) &\text{ requires } O(n_{F_1}) \text{ operations,} \\ \partial^2 F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x}) &\text{ requires } O(n_{F_2}) \text{ operations,} \end{aligned} \tag{D.7}$$

for $i \in \{1, 2, \ldots, n_x\}$ and $k \in \mathcal{I}_\Delta$. In computational complexity counting, I consider $O(n_g)$, $O(n_{g_1})$, $O(n_f)$, $O(n_{f_1})$, $O(n_{f_2})$, $O(n_F)$, $O(n_{F_1})$, $O(n_{F_2}) \geq O(1)$ and $n_{g_2} = 0$ or $O(n_{g_2}) \geq O(1)$.

D.1.3 Counting the Computational Complexity of $r(\mathbf{p}, \mathbf{x}; \lambda)$ Descent

Theorem 7. *An iteration of $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent requires*

$$\begin{aligned} &O\big(n_\sigma n_\Delta (n_g n_y + n_f n_x)\big) + O\big(n_\Delta n_p (n_{g_1} n_y + n_{f_1} n_x + n_{g_2} n_y + n_{f_2} n_x)\big) + \\ &O\big(n_\Delta n_x (n_{g_1} n_y + n_{f_1} n_x n_\delta + n_{g_2} n_y + n_{f_2} n_x n_\delta)\big) \end{aligned} \tag{D.8}$$

operations, with $O(n_\sigma)$ line-search test points.

Proof. In each iteration of descent, I calculate values of $r(\mathbf{p}, \mathbf{x}; \lambda)$. Calculating $r(\mathbf{p}, \mathbf{x}; \lambda)$, as in equation (D.1), is equivalent in computational complexity to calculating

$$\begin{aligned} d_{\mathrm{y}_{j,k}}\big(g_{j,k}(\mathbf{p}, \mathbf{x})\big) &\text{ for all } (j,k) \in \{1, \ldots, n_y\} \times \{1 \ldots, n_t\}, \\ d_{\hat{\mathrm{y}}_{j,k}}\big(g_{j,k}(\mathbf{p}, \mathbf{x})\big) &\text{ for all } (j,k) \in \{1, \ldots, n_y\} \times \mathcal{I}_{\hat{\mathrm{y}}}, \\ h_i(\mathbf{x}) &\text{ for all } i \in \{1, \ldots, n_x\}, \\ d_{\Delta_{i,k}}\big(f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})\big) &\text{ for all } (i,k) \in \{1, \ldots, n_x\} \times \mathcal{I}_\Delta, \end{aligned} \tag{D.9}$$

which, respectively, require

$$\begin{aligned} O(1) \circ O(n_g) \times n_y n_t &= O(n_g n_y n_t), \\ O(1) \circ O(n_g) \times n_y (n_\Delta - n_t) &= O\big(n_g n_y (n_\Delta - n_t)\big), \\ O(\leq n_f n_\Delta) \times n_x &= O(\leq n_f n_\Delta n_x), \\ O(1) \circ O(n_f) \times n_x n_\Delta &= O(n_f n_x n_\Delta) \end{aligned} \tag{D.10}$$

operations to calculate, as stipulated in Equations (D.3), (D.4), (D.5), and (D.6). Thus, in total,

calculating $r(\mathbf{p}, \mathbf{x}; \lambda)$ requires

$$O(n_g n_y n_t) + O\big(n_g n_y (n_\Delta - n_t)\big) + O(\leq n_f n_\Delta n_x) + O(n_f n_x n_\Delta) = \\ O(n_g n_y n_\Delta) + O(n_f n_x n_\Delta) = O\big(n_\Delta (n_g n_y + n_f n_x)\big) \tag{D.11}$$

operations.

In each iteration of descent, I calculate first order and un-mixed second order partial derivatives of $r(\mathbf{p}, \mathbf{x}; \lambda)$ with respect to all parameters.

$$\frac{\partial r(\mathbf{p}, \mathbf{x}; \lambda)}{\partial p_l} = \\ \frac{1-\lambda}{n_y} \sum_{j=1}^{n_y} \sum_{k=1}^{n_t} \frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}} \frac{\partial g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial p_l} + \frac{(1-\lambda)^2}{n_y} \sum_{j=1}^{n_y} \sum_{k \in \mathcal{I}_{\hat{\mathrm{y}}}} \frac{\partial d_{\hat{\mathrm{y}}_{j,k}}}{\partial g_{j,k}} \frac{\partial g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial p_l} + \\ \frac{\lambda}{n_x} \sum_{i=1}^{n_x} h_i(\mathbf{x}) \sum_{k \in \mathcal{I}_\Delta} \frac{\partial d_{\Delta_{i,k}}}{\partial f_{i,k}} \frac{\partial f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial p_l}, \tag{D.12}$$

$$\frac{\partial^2 r(\mathbf{p}, \mathbf{x}; \lambda)}{\partial p_l^2} = \\ \frac{1-\lambda}{n_y} \sum_{j=1}^{n_y} \sum_{k=1}^{n_t} \left(\frac{\partial^2 d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}^2} \Big(\frac{\partial g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial p_l} \Big)^2 + \frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}} \frac{\partial^2 g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial p_l^2} \right) + \\ \frac{(1-\lambda)^2}{n_y} \sum_{j=1}^{n_y} \sum_{k \in \mathcal{I}_{\hat{\mathrm{y}}}} \left(\frac{\partial^2 d_{\hat{\mathrm{y}}_{j,k}}}{\partial g_{j,k}^2} \Big(\frac{\partial g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial p_l} \Big)^2 + \frac{\partial d_{\hat{\mathrm{y}}_{j,k}}}{\partial g_{j,k}} \frac{\partial^2 g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial p_l^2} \right) + \\ \frac{\lambda}{n_x} \sum_{i=1}^{n_x} h_i(\mathbf{x}) \sum_{k \in \mathcal{I}_\Delta} \left(\frac{\partial^2 d_{\Delta_{i,k}}}{\partial f_{i,k}^2} \Big(\frac{\partial f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial p_l} \Big)^2 + \frac{\partial d_{\Delta_{i,k}}}{\partial f_{i,k}} \frac{\partial^2 f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial p_l^2} \right). \tag{D.13}$$

Calculating $\partial r(\mathbf{p}, \mathbf{x}; \lambda) / \partial p_l$, as in equation (D.12), for all $l \in \{1, 2, \ldots, n_p\}$ requires calculating

partial derivative values,

$$\begin{aligned}
\frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}} &\text{ for all } (j,k) \in \{1,\ldots,n_y\} \times \{1\ldots,n_t\},\\
\frac{\partial g_{j,k}(\mathbf{p},\mathbf{x})}{\partial p_l} &\text{ for all } (j,k,l) \in \{1,\ldots,n_y\} \times \{1\ldots,n_t\} \times \{1\ldots,n_p\},\\
\frac{\partial d_{\hat{\mathrm{y}}_{j,k}}}{\partial g_{j,k}} &\text{ for all } (j,k) \in \{1,\ldots,n_y\} \times \mathcal{I}_{\hat{\mathrm{y}}},\\
\frac{\partial g_{j,k}(\mathbf{p},\mathbf{x})}{\partial p_l} &\text{ for all } (j,k,l) \in \{1,\ldots,n_y\} \times \mathcal{I}_{\hat{\mathrm{y}}} \times \{1\ldots,n_p\},\\
\frac{\partial d_{\Delta_{i,k}}}{\partial f_{i,k}} &\text{ for all } (i,k) \in \{1,\ldots,n_x\} \times \mathcal{I}_{\Delta},\\
\frac{\partial f_{i,k}(\mathbf{t},\mathbf{p},\mathbf{x})}{\partial p_l} &\text{ for all } (i,k,l) \in \{1,\ldots,n_x\} \times \mathcal{I}_{\Delta} \times \{1\ldots,n_p\},
\end{aligned} \tag{D.14}$$

which, respectively, require

$$\begin{aligned}
O(1) \circ O(I_0) \times n_y n_t &= O(n_y n_t),\\
O(n_{g_1}) \times n_y n_t n_p &= O(n_{g_1} n_y n_t n_p),\\
O(1) \circ O(I_0) \times n_y (n_\Delta - n_t) &= O\big(n_y (n_\Delta - n_t)\big),\\
O(n_{g_1}) \times n_y (n_\Delta - n_t) n_p &= O\big(n_{g_1} n_y (n_\Delta - n_t) n_p\big),\\
O(1) \circ O(I_0) \times n_x n_\Delta &= O(n_x n_\Delta),\\
O(n_{f_1}) \times n_x n_\Delta n_p &= O(n_{f_1} n_x n_\Delta n_p),
\end{aligned} \tag{D.15}$$

operations to calculate, as stipulated in Equations (D.3), (D.4), and (D.5), where I_0 indicates values that have been calculated previously and $O(I_0) = 1$. Apart from calculating partial derivative values, calculating $\partial r(\mathbf{p}, \mathbf{x}; \lambda)/\partial p_l$, as in equation (D.12), for all $l \in \{1, 2, \ldots, n_p\}$ is equivalent in computational complexity to calculating

$$\begin{aligned}
\frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}} \frac{\partial g_{j,k}(\mathbf{p},\mathbf{x})}{\partial p_l} &\text{ for all } (j,k,l) \in \{1,\ldots,n_y\} \times \{1\ldots,n_t\} \times \{1\ldots,n_p\},\\
\frac{\partial d_{\hat{\mathrm{y}}_{j,k}}}{\partial g_{j,k}} \frac{\partial g_{j,k}(\mathbf{p},\mathbf{x})}{\partial p_l} &\text{ for all } (j,k,l) \in \{1,\ldots,n_y\} \times \mathcal{I}_{\hat{\mathrm{y}}} \times \{1\ldots,n_p\},\\
\frac{\partial d_{\Delta_{i,k}}}{\partial f_{i,k}} \frac{\partial f_{i,k}(\mathbf{t},\mathbf{p},\mathbf{x})}{\partial p_l} &\text{ for all } (i,k,l) \in \{1,\ldots,n_x\} \times \mathcal{I}_{\Delta} \times \{1\ldots,n_p\},\\
h_i(\mathbf{x}) \cdot I_0 &\text{ for all } (i,l) \in \{1,\ldots,n_x\} \times \{1\ldots,n_p\},
\end{aligned} \tag{D.16}$$

which, respectively, require

$$\begin{aligned} O(1) \circ O(I_0) \times n_y n_t n_p &= O(n_y n_t n_p), \\ O(1) \circ O(I_0) \times n_y (n_\Delta - n_t) n_p &= O\big(n_y (n_\Delta - n_t) n_p\big), \\ O(1) \circ O(I_0) \times n_x n_\Delta n_p &= O(n_x n_\Delta n_p), \\ O(1) \circ O(I_0) \times n_x n_p &= O(n_x n_p) \end{aligned} \tag{D.17}$$

operations to calculate. Thus, from computational complexity counts (D.15) and (D.17), calculating $\partial r(\mathbf{p}, \mathbf{x}; \lambda)/\partial p_l$ for all $l \in \{1, 2, \ldots, n_p\}$ requires

$$\begin{aligned} &O(n_y n_t) + O(n_{g_1} n_y n_t n_p) + O\big(n_y (n_\Delta - n_t)\big) + O\big(n_{g_1} n_y (n_\Delta - n_t) n_p\big) + \\ &O(n_x n_\Delta) + O(n_{f_1} n_x n_\Delta n_p) + O(n_y n_t n_p) + O\big(n_y (n_\Delta - n_t) n_p\big) + O(n_x n_\Delta n_p) + \\ &O(n_x n_p) = O\big(n_\Delta n_p (n_{g_1} n_y + n_{f_1} n_x)\big) \end{aligned} \tag{D.18}$$

operations.

Calculating $\partial^2 r(\mathbf{p}, \mathbf{x}; \lambda)/\partial p_l^2$, as in equation (D.13), for all $l \in \{1, 2, \ldots, n_p\}$ requires calculating partial derivative values,

$$\begin{aligned} &\frac{\partial^2 d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}^2} \text{ for all } (j,k) \in \{1, \ldots, n_y\} \times \{1 \ldots, n_t\}, \\ &\frac{\partial^2 g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial p_l^2} \text{ for all } (j,k,l) \in \{1, \ldots, n_y\} \times \{1 \ldots, n_t\} \times \{1 \ldots, n_p\}, \\ &\frac{\partial^2 d_{\hat{\mathrm{y}}_{j,k}}}{\partial g_{j,k}^2} \text{ for all } (j,k) \in \{1, \ldots, n_y\} \times \mathcal{I}_{\hat{\mathrm{y}}}, \\ &\frac{\partial^2 g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial p_l^2} \text{ for all } (j,k,l) \in \{1, \ldots, n_y\} \times \mathcal{I}_{\hat{\mathrm{y}}} \times \{1 \ldots, n_p\}, \\ &\frac{\partial^2 d_{\Delta_{i,k}}}{\partial f_{i,k}^2} \text{ for all } (i,k) \in \{1, \ldots, n_x\} \times \mathcal{I}_\Delta, \\ &\frac{\partial^2 f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial p_l^2} \text{ for all } (i,k,l) \in \{1, \ldots, n_x\} \times \mathcal{I}_\Delta \times \{1 \ldots, n_p\}, \end{aligned} \tag{D.19}$$

which, respectively, require

$$\begin{aligned} O(1) \circ O(I_0) \times n_y n_t &= O(n_y n_t), \\ O(n_{g_2}) \times n_y n_t n_p &= O(n_{g_2} n_y n_t n_p), \\ O(1) \circ O(I_0) \times n_y (n_\Delta - n_t) &= O\big(n_y (n_\Delta - n_t)\big), \\ O(n_{g_2}) \times n_y (n_\Delta - n_t) n_p &= O\big(n_{g_2} n_y (n_\Delta - n_t) n_p\big), \\ O(1) \circ O(I_0) \times n_x n_\Delta &= O(n_x n_\Delta), \\ O(n_{f_2}) \times n_x n_\Delta n_p &= O(n_{f_2} n_x n_\Delta n_p), \end{aligned} \tag{D.20}$$

operations to calculate, as stipulated in Equations (D.3), (D.4), and (D.5). Apart from calculating partial derivative values, calculating $\partial^2 r(\mathbf{p}, \mathbf{x}; \lambda)/\partial p_l^2$, as in equation (D.13), for all $l \in \{1, 2, \ldots, n_p\}$ is equivalent in computational complexity to calculating

$$\begin{gathered}
\frac{\partial^2 d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}^2}\Big(\frac{\partial g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial p_l}\Big)^2 \text{ for all } (j,k,l) \in \{1,\ldots,n_y\} \times \{1\ldots,n_t\} \times \{1\ldots,n_p\}, \\
\frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}} \frac{\partial^2 g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial p_l^2} \text{ for all } (j,k,l) \in \{1,\ldots,n_y\} \times \{1\ldots,n_t\} \times \{1\ldots,n_p\}, \\
\frac{\partial^2 d_{\hat{\mathrm{y}}_{j,k}}}{\partial g_{j,k}^2}\Big(\frac{\partial g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial p_l}\Big)^2 \text{ for all } (j,k,l) \in \{1,\ldots,n_y\} \times \mathcal{I}_{\hat{\mathrm{y}}} \times \{1\ldots,n_p\}, \\
\frac{\partial d_{\hat{\mathrm{y}}_{j,k}}}{\partial g_{j,k}} \frac{\partial^2 g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial p_l^2} \text{ for all } (j,k,l) \in \{1,\ldots,n_y\} \times \mathcal{I}_{\hat{\mathrm{y}}} \times \{1\ldots,n_p\}, \\
\frac{\partial^2 d_{\Delta_{i,k}}}{\partial f_{i,k}^2}\Big(\frac{\partial f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial p_l}\Big)^2 \text{ for all } (i,k,l) \in \{1,\ldots,n_x\} \times \mathcal{I}_{\Delta} \times \{1\ldots,n_p\}, \\
\frac{\partial d_{\Delta_{i,k}}}{\partial f_{i,k}} \frac{\partial^2 f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial p_l^2} \text{ for all } (i,k,l) \in \{1,\ldots,n_x\} \times \mathcal{I}_{\Delta} \times \{1\ldots,n_p\}, \\
h_i(\mathbf{x}) \cdot I_0 \text{ for all } (i,l) \in \{1,\ldots,n_x\} \times \{1\ldots,n_p\},
\end{gathered} \tag{D.21}$$

which, respectively, require

$$\begin{aligned}
O(1) \circ O(I_0) \times n_y n_t n_p &= O(n_y n_t n_p), \\
I_{g_2} \cdot O(1) \circ O(I_0) \times n_y n_t n_p &= O(I_{g_2} n_y n_t n_p), \\
O(1) \circ O(I_0) \times n_y (n_\Delta - n_t) n_p &= O\big(n_y (n_\Delta - n_t) n_p\big), \\
O(1) \circ O(I_0) \times n_y (n_\Delta - n_t) n_p &= O\big(n_y (n_\Delta - n_t) n_p\big), \\
O(1) \circ O(I_0) \times n_x n_\Delta n_p &= O(n_x n_\Delta n_p), \\
O(1) \circ O(I_0) \times n_x n_\Delta n_p &= O(n_x n_\Delta n_p), \\
O(1) \circ O(I_0) \times n_x n_p &= O(n_x n_p)
\end{aligned} \tag{D.22}$$

operations to calculate, where

$$I_{g_2} = \begin{cases} 0 & \text{if } n_{g_2} = 0 \\ 1 & \text{if } O(n_{g_2}) \geq 1 \end{cases}. \tag{D.23}$$

Thus, from complexity counts (D.20) and (D.22), calculating $\partial^2 r(\mathbf{p}, \mathbf{x}; \lambda)/\partial p_l^2$ for all $l \in$

$\{1, 2, \ldots, n_p\}$ requires

$$O(n_y n_t) + O(n_{g_2} n_y n_t n_p) + O\big(n_y(n_\Delta - n_t)\big) + O\big(n_{g_2} n_y (n_\Delta - n_t) n_p\big) + O(n_x n_\Delta) + O(n_{f_2} n_x n_\Delta n_p) + O(n_y n_t n_p) + O(I_{g_2} n_y n_t n_p) + O\big(n_y(n_\Delta - n_t)n_p\big) + O\big(n_y(n_\Delta - n_t)n_p\big) + O(n_x n_\Delta n_p) + O(n_x n_\Delta n_p) + O(n_x n_p) = O(n_{g_2} n_y n_\Delta n_p) + O(n_{f_2} n_x n_\Delta n_p) + O(n_y n_\Delta n_p) = O\big(n_\Delta n_p (n_{g_2} n_y + n_{f_2} n_x)\big) \tag{D.24}$$

operations. I note that computational complexity count (D.24) holds when $n_{g_2} = 0$, as $n_y \leq n_x$.

In each iteration of descent, I calculate first order and un-mixed second order partial derivatives of $r(\mathbf{p}, \mathbf{x}; \lambda)$ with respect to all state values.

$$\frac{\partial r(\mathbf{p}, \mathbf{x}; \lambda)}{\partial x_{l,m}} = \frac{1-\lambda}{n_y} \sum_{j=1}^{n_y} \sum_{k=1}^{n_t} \frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}} \frac{\partial g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}} + \frac{(1-\lambda)^2}{n_y} \sum_{j=1}^{n_y} \sum_{k \in \mathcal{I}_{\hat{\mathrm{y}}}} \frac{\partial d_{\hat{\mathrm{y}}_{j,k}}}{\partial g_{j,k}} \frac{\partial g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}} + \frac{\lambda}{n_x} \sum_{i=1}^{n_x} h_i(\mathbf{x}) \sum_{k \in \mathcal{I}_\Delta} \frac{\partial d_{\Delta_{i,k}}}{\partial f_{i,k}} \frac{\partial f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial x_{l,m}} + \frac{\lambda}{n_x} \sum_{i=1}^{n_x} \frac{\partial h_i(\mathbf{x})}{\partial x_{l,m}} \sum_{k \in \mathcal{I}_\Delta} d_{\Delta_{i,k}}\big(f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})\big),$$

$$\frac{\partial^2 r(\mathbf{p}, \mathbf{x}; \lambda)}{\partial x_{l,m}^2} = \frac{1-\lambda}{n_y} \sum_{j=1}^{n_y} \sum_{k=1}^{n_t} \left(\frac{\partial^2 d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}^2} \Big(\frac{\partial g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}} \Big)^2 + \frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}} \frac{\partial^2 g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}^2} \right) + \frac{(1-\lambda)^2}{n_y} \sum_{j=1}^{n_y} \sum_{k \in \mathcal{I}_{\hat{\mathrm{y}}}} \left(\frac{\partial^2 d_{\hat{\mathrm{y}}_{j,k}}}{\partial g_{j,k}^2} \Big(\frac{\partial g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}} \Big)^2 + \frac{\partial d_{\hat{\mathrm{y}}_{j,k}}}{\partial g_{j,k}} \frac{\partial^2 g_{j,k}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}^2} \right) + \frac{\lambda}{n_x} \sum_{i=1}^{n_x} h_i(\mathbf{x}) \sum_{k \in \mathcal{I}_\Delta} \left(\frac{\partial^2 d_{\Delta_{i,k}}}{\partial f_{i,k}^2} \Big(\frac{\partial f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial x_{l,m}} \Big)^2 + \frac{\partial d_{\Delta_{i,k}}}{\partial f_{i,k}} \frac{\partial^2 f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial x_{l,m}^2} \right) + \frac{\lambda}{n_x} \sum_{i=1}^{n_x} \left(2 \frac{\partial h_i(\mathbf{x})}{\partial x_{l,m}} \sum_{k \in \mathcal{I}_\Delta} \frac{\partial d_{\Delta_{i,k}}}{\partial f_{i,k}} \frac{\partial f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial x_{l,m}} + \frac{\partial^2 h_i(\mathbf{x})}{\partial x_{l,m}^2} \sum_{k \in \mathcal{I}_\Delta} d_{\Delta_{i,k}}\big(f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})\big) \right). \tag{D.25}$$

Observable-state functions, $g_{j,k}$, depend only on the state values at grid index k; auxiliary functions, $h_i(\mathbf{x})$, depend only on state values in the i^{th} state; and $f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$ depend on $x_{l,m}$

for only a small fraction of k in $\mathcal{I}_\Delta$, at k in $\mathcal{I}_{\Delta_m} \subset \mathcal{I}_\Delta$. Thus,

$$\frac{\partial r(\mathbf{p}, \mathbf{x}; \lambda)}{\partial x_{l,m}} =$$
$$\frac{1-\lambda}{n_y} \sum_{j=1}^{n_y} \frac{\partial d_{\mathrm{y}_{j,m}}}{\partial g_{j,m}} \frac{\partial g_{j,m}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}} + \frac{(1-\lambda)^2}{n_y} \sum_{j=1}^{n_y} \frac{\partial d_{\hat{\mathrm{y}}_{j,m}}}{\partial g_{j,m}} \frac{\partial g_{j,m}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}} +$$
$$\frac{\lambda}{n_x} \sum_{i=1}^{n_x} h_i(\mathbf{x}) \sum_{k \in \mathcal{I}_{\Delta_m}} \frac{\partial d_{\Delta_{i,k}}}{\partial f_{i,k}} \frac{\partial f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial x_{l,m}} + \frac{\lambda}{n_x} \frac{\partial h_l(\mathbf{x})}{\partial x_{l,m}} \sum_{k \in \mathcal{I}_\Delta} d_{\Delta_{i,k}}\big(f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})\big), \tag{D.26}$$

$$\frac{\partial^2 r(\mathbf{p}, \mathbf{x}; \lambda)}{\partial x_{l,m}^2} =$$
$$\frac{1-\lambda}{n_y} \sum_{j=1}^{n_y} \left(\frac{\partial^2 d_{\mathrm{y}_{j,m}}}{\partial g_{j,m}^2} \Big(\frac{\partial g_{j,m}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}} \Big)^2 + \frac{\partial d_{\mathrm{y}_{j,m}}}{\partial g_{j,m}} \frac{\partial^2 g_{j,m}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}^2} \right) +$$
$$\frac{(1-\lambda)^2}{n_y} \sum_{j=1}^{n_y} \left(\frac{\partial^2 d_{\hat{\mathrm{y}}_{j,m}}}{\partial g_{j,m}^2} \Big(\frac{\partial g_{j,m}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}} \Big)^2 + \frac{\partial d_{\hat{\mathrm{y}}_{j,m}}}{\partial g_{j,m}} \frac{\partial^2 g_{j,m}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}^2} \right) +$$
$$\frac{\lambda}{n_x} \sum_{i=1}^{n_x} h_i(\mathbf{x}) \sum_{k \in \mathcal{I}_{\Delta_m}} \left(\frac{\partial^2 d_{\Delta_{i,k}}}{\partial f_{i,k}^2} \Big(\frac{\partial f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial x_{l,m}} \Big)^2 + \frac{\partial d_{\Delta_{i,k}}}{\partial f_{i,k}} \frac{\partial^2 f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial x_{l,m}^2} \right) +$$
$$\frac{\lambda}{n_x} \left(2 \frac{\partial h_l(\mathbf{x})}{\partial x_{l,m}} \sum_{k \in \mathcal{I}_{\Delta_m}} \frac{\partial d_{\Delta_{i,k}}}{\partial f_{i,k}} \frac{\partial f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial x_{l,m}} + \frac{\partial^2 h_l(\mathbf{x})}{\partial x_{l,m}^2} \sum_{k \in \mathcal{I}_\Delta} d_{\Delta_{i,k}}\big(f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})\big) \right). \tag{D.27}$$

Calculating $\partial r(\mathbf{p}, \mathbf{x}; \lambda)/\partial x_{l,m}$, as in equation (D.26), for all $l \in \{1, 2, \ldots, n_x\}$ and all m in $\mathcal{I}_\Delta$ requires calculating partial derivative values,

$$\begin{gathered} \frac{\partial g_{j,m}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}} \text{ for all } (j, l, m) \in \{1, \ldots, n_y\} \times \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \\ \frac{\partial f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial x_{l,m}} \text{ for all } (i, k, l, m) \in \{1, \ldots, n_x\} \times \mathcal{I}_{\Delta_m} \times \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \\ \frac{\partial h_l(\mathbf{x})}{\partial x_{l,m}} \text{ for all } (l, m) \in \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \end{gathered} \tag{D.28}$$

which, respectively, require

$$\begin{aligned} O(n_{g_1}) \times n_y n_x n_\Delta &= O(n_{g_1} n_y n_x n_\Delta), \\ O(n_{f_1}) \times n_x n_\delta n_x n_\Delta &= O(n_{f_1} n_x n_\delta n_x n_\Delta), \\ O(\leq n_{f_1} n_\delta) \times n_x n_\Delta &= O(\leq n_{f_1} n_\delta n_x n_\Delta) \end{aligned} \tag{D.29}$$

operations to calculate, as stipulated in Equations (D.4), (D.5), and (D.6). Apart from calculating partial derivative values, calculating $\partial r(\mathbf{p}, \mathbf{x}; \lambda)/\partial x_{l,m}$, as in equation (D.26), for all

$l \in \{1, 2, \ldots, n_x\}$ and all m in $\mathcal{I}_\Delta$ is equivalent in computational complexity to calculating

$$\begin{gathered}
\frac{\partial d_{\mathrm{y}_{j,m}}}{\partial g_{j,m}} \frac{\partial g_{j,m}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}} \text{ for all } (j, l, m) \in \{1, \ldots, n_y\} \times \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \\
\frac{\partial d_{\hat{\mathrm{y}}_{j,m}}}{\partial g_{j,m}} \frac{\partial g_{j,m}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}} \text{ for all } (j, l, m) \in \{1, \ldots, n_y\} \times \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \\
\frac{\partial d_{\Delta_{i,k}}}{\partial f_{i,k}} \frac{\partial f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial x_{l,m}} \text{ for all } (i, k, l, m) \in \{1, \ldots, n_x\} \times \mathcal{I}_{\Delta_m} \times \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \\
h_i(\mathbf{x}) \cdot I_0 \text{ for all } (i, l, m) \in \{1, \ldots, n_x\} \times \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \\
\frac{\partial h_l(\mathbf{x})}{\partial x_{l,m}} \cdot I_0 \text{ for all } (l, m) \in \{1 \ldots, n_x\} \times \mathcal{I}_\Delta,
\end{gathered} \tag{D.30}$$

which, respectively, require

$$\begin{aligned}
O(1) \circ O(I_0) \times n_y n_x n_\Delta &= O(n_y n_x n_\Delta), \\
O(1) \circ O(I_0) \times n_y n_x n_\Delta &= O(n_y n_x n_\Delta), \\
O(1) \circ O(I_0) \times n_x n_\delta n_x n_\Delta &= O(n_x n_\delta n_x n_\Delta), \\
O(1) \circ O(I_0) \times n_x n_x n_\Delta &= O(n_x n_x n_\Delta), \\
O(1) \circ O(I_0) \times n_x n_\Delta &= O(n_x n_\Delta)
\end{aligned} \tag{D.31}$$

operations to calculate. Thus, from complexity counts (D.29) and (D.31), calculating $\partial r(\mathbf{p}, \mathbf{x}; \lambda)/\partial x_{l,m}$ for all $l \in \{1, 2, \ldots, n_x\}$ and all m in $\mathcal{I}_\Delta$ requires

$$\begin{gathered}
O(n_{g_1} n_y n_x n_\Delta) + O(n_{f_1} n_x n_\delta n_x n_\Delta) + O(\leq n_{f_1} n_\delta n_x n_\Delta) + O(n_y n_x n_\Delta) + \\
O(n_y n_x n_\Delta) + O(n_x n_\delta n_x n_\Delta) + O(n_x n_x n_\Delta) + O(n_x n_\Delta) = \\
O\big(n_\Delta n_x (n_{g_1} n_y + n_{f_1} n_x n_\delta)\big)
\end{gathered} \tag{D.32}$$

operations. Calculating $\partial^2 r(\mathbf{p}, \mathbf{x}; \lambda)/\partial x_{l,m}^2$, as in equation (D.27), for all $l \in \{1, 2, \ldots, n_x\}$ and all m in $\mathcal{I}_\Delta$ requires calculating partial derivative values,

$$\begin{gathered}
\frac{\partial^2 g_{j,m}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}^2} \text{ for all } (j, l, m) \in \{1, \ldots, n_y\} \times \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \\
\frac{\partial^2 f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial x_{l,m}^2} \text{ for all } (i, k, l, m) \in \{1, \ldots, n_x\} \times \mathcal{I}_{\Delta_m} \times \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \\
\frac{\partial^2 h_l(\mathbf{x})}{\partial x_{l,m}^2} \text{ for all } (l, m) \in \{1 \ldots, n_x\} \times \mathcal{I}_\Delta,
\end{gathered} \tag{D.33}$$

which, respectively, require

$$\begin{aligned} O(n_{g_2}) \times n_y n_x n_\Delta &= O(n_{g_2} n_y n_x n_\Delta), \\ O(n_{f_2}) \times n_x n_\delta n_x n_\Delta &= O(n_{f_2} n_x n_\delta n_x n_\Delta), \\ O(\leq n_{f_2} n_\delta) \times n_x n_\Delta &= O(\leq n_{f_2} n_\delta n_x n_\Delta) \end{aligned} \tag{D.34}$$

operations to calculate, as stipulated in Equations (D.4), (D.5), and (D.6). Apart from calculating partial derivative values, calculating $\partial^2 r(\mathbf{p}, \mathbf{x}; \lambda)/\partial x_{l,m}^2$, as in equation (D.27), for all $l \in \{1, 2, \ldots, n_x\}$ and all m in $\mathcal{I}_\Delta$ is equivalent in computational complexity to calculating

$$\begin{aligned} \frac{\partial^2 d_{\mathrm{y}_{j,m}}}{\partial g_{j,m}^2} \Big(\frac{\partial g_{j,m}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}} \Big)^2 &\text{ for all } (j, l, m) \in \{1, \ldots, n_y\} \times \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \\ \frac{\partial d_{\mathrm{y}_{j,m}}}{\partial g_{j,m}} \frac{\partial^2 g_{j,m}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}^2} &\text{ for all } (j, l, m) \in \{1, \ldots, n_y\} \times \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \\ \frac{\partial^2 d_{\hat{\mathrm{y}}_{j,m}}}{\partial g_{j,m}^2} \Big(\frac{\partial g_{j,m}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}} \Big)^2 &\text{ for all } (j, l, m) \in \{1, \ldots, n_y\} \times \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \\ \frac{\partial d_{\hat{\mathrm{y}}_{j,m}}}{\partial g_{j,m}} \frac{\partial^2 g_{j,m}(\mathbf{p}, \mathbf{x})}{\partial x_{l,m}^2} &\text{ for all } (j, l, m) \in \{1, \ldots, n_y\} \times \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \\ \frac{\partial^2 d_{\Delta_{i,k}}}{\partial f_{i,k}^2} \Big(\frac{\partial f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial x_{l,m}} \Big)^2 &\text{ for all } (i, k, l, m) \in \{1, \ldots, n_x\} \times \mathcal{I}_{\Delta_m} \times \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \\ \frac{\partial d_{\Delta_{i,k}}}{\partial f_{i,k}} \frac{\partial^2 f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})}{\partial x_{l,m}^2} &\text{ for all } (i, k, l, m) \in \{1, \ldots, n_x\} \times \mathcal{I}_{\Delta_m} \times \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \\ h_i(\mathbf{x}) \cdot I_0 &\text{ for all } (i, l, m) \in \{1, \ldots, n_x\} \times \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \\ \frac{\partial h_l(\mathbf{x})}{\partial x_{l,m}} \cdot I_0 &\text{ for all } (l, m) \in \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \\ \frac{\partial^2 h_l(\mathbf{x})}{\partial x_{l,m}^2} \cdot I_0 &\text{ for all } (l, m) \in \{1 \ldots, n_x\} \times \mathcal{I}_\Delta, \end{aligned} \tag{D.35}$$

which, respectively, require

$$\begin{gathered}
O(1) \circ O(I_0) \times n_y n_x n_\Delta = O(n_y n_x n_\Delta), \\
I_{g_2} \cdot O(1) \circ O(I_0) \times n_y n_x n_\Delta = O(I_{g_2} n_y n_x n_\Delta), \\
O(1) \circ O(I_0) \times n_y n_x n_\Delta = O(n_y n_x n_\Delta), \\
I_{g_2} \cdot O(1) \circ O(I_0) \times n_y n_x n_\Delta = O(I_{g_2} n_y n_x n_\Delta), \\
O(1) \circ O(I_0) \times n_x n_\delta n_x n_\Delta = O(n_x n_\delta n_x n_\Delta), \\
O(1) \circ O(I_0) \times n_x n_\delta n_x n_\Delta = O(n_x n_\delta n_x n_\Delta), \\
O(1) \circ O(I_0) \times n_x n_x n_\Delta = O(n_x n_x n_\Delta), \\
O(1) \circ O(I_0) \times n_x n_\Delta = O(n_x n_\Delta), \\
O(1) \circ O(I_0) \times n_x n_\Delta = O(n_x n_\Delta)
\end{gathered} \tag{D.36}$$

operations to calculate. Thus, from complexity counts (D.34) and (D.36), calculating $\partial^2 r(\mathbf{p}, \mathbf{x}; \lambda)/\partial x_{l,m}^2$ for all $l \in \{1, 2, \ldots, n_x\}$ and all m in $\mathcal{I}_\Delta$ requires

$$\begin{gathered}
O(n_{g_2} n_y n_x n_\Delta) + O(n_{f_2} n_x n_\delta n_x n_\Delta) + O(\leq n_{f_2} n_\delta n_x n_\Delta) + O(n_y n_x n_\Delta) + \\
O(I_{g_2} n_y n_x n_\Delta) + O(n_y n_x n_\Delta) + O(I_{g_2} n_y n_x n_\Delta) + O(n_x n_\delta n_x n_\Delta) + \\
O(n_x n_\delta n_x n_\Delta) + O(n_x n_x n_\Delta) + O(n_x n_\Delta) + O(n_x n_\Delta) = \\
O(n_{g_2} n_y n_x n_\Delta) + O(n_{f_2} n_x n_\delta n_x n_\Delta) + O(n_y n_x n_\Delta) = \\
O\big(n_\Delta n_x (n_{g_2} n_y + n_{f_2} n_x n_\delta)\big)
\end{gathered} \tag{D.37}$$

operations. I note that computational complexity count (D.37) holds when $n_{g_2} = 0$, as $n_y \leq n_x$.

In each iteration of descent, I generate $O(n_\sigma)$ line-search test points, one for each value of σ_j. Generating each test point requires an update of all parameters and state values, requiring $O(n_p + n_x n_\Delta)$ operations. I consider $r(\mathbf{p}, \mathbf{x}; \lambda)$ with fewer parameter values than state values. Thus, $n_p < n_x n_\Delta$, which implies that $O(n_p + n_x n_\Delta) = O(n_x n_\Delta)$. From complexity count (D.11), calculating $r(\mathbf{p}, \mathbf{x}; \lambda)$ at each line-search test point requires $O(n_\Delta(n_g n_y + n_f n_x))$ operations. Thus, updating all parameters and state values at all line-search test points, calculating $r(\mathbf{p}, \mathbf{x}; \lambda)$ at all line-search test points, calculating first order and un-mixed second order partial derivatives of $r(\mathbf{p}, \mathbf{x}; \lambda)$ with respect to all parameters, and calculating first order and un-mixed second order partial derivatives of $r(\mathbf{p}, \mathbf{x}; \lambda)$ with respect to all state values requires

$$\begin{gathered}
O(n_\sigma n_x n_\Delta) + O\big(n_\sigma n_\Delta (n_g n_y + n_f n_x)\big) + \\
O\big(n_\Delta n_p (n_{g_1} n_y + n_{f_1} n_x)\big) + O\big(n_\Delta n_p (n_{g_2} n_y + n_{f_2} n_x)\big) + \\
O\big(n_\Delta n_x (n_{g_1} n_y + n_{f_1} n_x n_\delta)\big) + O\big(n_\Delta n_x (n_{g_2} n_y + n_{f_2} n_x n_\delta)\big) = \\
O\big(n_\sigma n_\Delta (n_g n_y + n_f n_x)\big) + O\big(n_\Delta n_p (n_{g_1} n_y + n_{f_1} n_x + n_{g_2} n_y + n_{f_2} n_x)\big) + \\
O\big(n_\Delta n_x (n_{g_1} n_y + n_{f_1} n_x n_\delta + n_{g_2} n_y + n_{f_2} n_x n_\delta)\big)
\end{gathered} \tag{D.38}$$

operations, with partial derivative complexity counts from (D.18), (D.24), (D.32), and (D.37). Therefore, an iteration of $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent requires

$$O\big(n_\sigma n_\Delta (n_g n_y + n_f n_x)\big) + O\big(n_\Delta n_p (n_{g_1} n_y + n_{f_1} n_x + n_{g_2} n_y + n_{f_2} n_x)\big) + O\big(n_\Delta n_x (n_{g_1} n_y + n_{f_1} n_x n_\delta + n_{g_2} n_y + n_{f_2} n_x n_\delta)\big) \tag{D.39}$$

operations. □

D.2 Computational Complexities of Numerical-Integration-Based Methods

Comparatively, I count the computational complexity required to minimize $r(\mathbf{q})$, where

$$r(\mathbf{q}) = \frac{1}{n_y} \sum_{j=1}^{n_y} \sum_{k=1}^{n_t} d_{\mathrm{y}_{j,k}}\big(g_{j,k}(\mathbf{q}, \mathbf{x})\big) :$$
$$f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0 \text{ for all } i \in \{1, 2, \ldots, n_x\} \text{ and for all } k \in \mathcal{I}_\Delta, \tag{D.40}$$

with $\mathbf{q}$ consisting of n_q elements, $q_1, q_2, \ldots, q_{n_q}$, which correspond to model parameters $p_1, p_2, \ldots, p_{n_p}$ and variable initial conditions and boundary values, and where $g_{j,k}(\mathbf{p}, \mathbf{x}) = \mathrm{y}_{j,k} - g_j(\mathbf{p}, x_{1,k}, \ldots, x_{n_x,k})$ with difference measure $d_{\mathrm{y}_{j,k}}$. As in counting the computation complexity required for an iteration of $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent, I define quantities for counting computational complexities as in Section D.1.2.

D.2.1 Counting the Computational Complexity of $r(\mathbf{q})$ Descent

Theorem 8. *For an explicit numerical solution method, an iteration of $r(\mathbf{q})$ descent requires*

$$O\big(n_\sigma (n_f n_x n_\Delta + n_g n_y n_t) + n_y n_t n_{g_1} (n_x + n_q)\big) + O\big(n_y n_t n_{g_2} (n_x n_x + n_q n_x)\big) + O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_x n_x n_q)\big) \tag{D.41}$$

operations, with $O(n_\sigma)$ line-search test points. For an implicit numerical solution method, an iteration of $r(\mathbf{q})$ descent requires

$$O\big(n_\sigma n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_\sigma n_g n_y n_t\big) + O\big(n_M (n_q + n_\sigma n_N) + n_y n_t n_{g_1} (n_x + n_q)\big) + O\big(n_y n_t n_{g_2} (n_x n_x + n_q n_x)\big) + O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_x n_x n_q)\big) \tag{D.42}$$

operations, with $O(n_M)$ operations in solving matrix equations for each of $O(n_N)$ iterations of Newton's method applied to $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$, for all $i \in \{1, 2, \ldots, n_x\}$ and $k \in \mathcal{I}_\Delta$.

Alternatively, for an explicit numerical solution method, an iteration of $r(\mathbf{q})$ descent with

partial derivative approximation by finite difference requires

$$O\big((n_\sigma + n_q)(n_f n_x n_\Delta + n_g n_y n_t)\big) \tag{D.43}$$

operations, and, for an implicit numerical solution method, an iteration of $r(\mathbf{q})$ *descent with partial derivative approximation by finite difference requires*

$$O\Big(\big(n_\sigma + n_q\big)\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big)\Big) \tag{D.44}$$

operations.

Proof. In each iteration of descent, I calculate values of $r(\mathbf{q})$. For an explicit numerical solution method, $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ is an explicit system of equations in $\mathbf{x}$, and solving $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ simply requires evaluating $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x})$. Thus, to determine $\mathbf{x}$, solving

$$f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0 \text{ for all } (i,k) \in \{1, \dots, n_x\} \times \mathcal{I}_\Delta \tag{D.45}$$

requires

$$O(n_f) \times n_x n_\Delta = O(n_f n_x n_\Delta) \tag{D.46}$$

operations, as stipulated in equation (D.5). After calculating $\mathbf{x}$, calculating $r(\mathbf{q})$ requires calculating

$$\frac{1}{n_y} \sum_{j=1}^{n_y} \sum_{k=1}^{n_t} d_{\mathrm{y}_{j,k}}\big(g_{j,k}(\mathbf{q}, \mathbf{x})\big), \tag{D.47}$$

which is equivalent in computational complexity to calculating

$$d_{\mathrm{y}_{j,k}}\big(g_{j,k}(\mathbf{q}, \mathbf{x})\big) \text{ for all } (j,k) \in \{1, \dots, n_y\} \times \{1, \dots, n_t\}, \tag{D.48}$$

which requires

$$O(1) \circ O(n_g) \times n_y n_t = O(n_g n_y n_t) \tag{D.49}$$

operations to calculate, as stipulated in Equations (D.3) and (D.4). Thus, from complexity counts (D.46) and (D.49), calculating $r(\mathbf{q})$ requires

$$O(n_f n_x n_\Delta) + O(n_g n_y n_t) = O(n_f n_x n_\Delta + n_g n_y n_t) \tag{D.50}$$

operations with an explicit numerical solution method.

In each iteration of descent, I calculate first order and un-mixed second order partial

derivatives of $r(\mathbf{q})$ with respect to all parameters,

$$\frac{\partial r(\mathbf{q})}{\partial q_l} = \frac{1}{n_y}\sum_{j=1}^{n_y}\sum_{k=1}^{n_t}\frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}}\left(\sum_{m=1}^{n_x}\frac{\partial g_{j,k}}{\partial x_{m,k}}\frac{\partial x_{m,k}}{\partial q_l}+\frac{\partial g_{j,k}}{\partial q_l}\right), \tag{D.51a}$$

$$\begin{aligned}\frac{\partial^2 r(\mathbf{q})}{\partial q_l^2} = \frac{1}{n_y}\sum_{j=1}^{n_y}\sum_{k=1}^{n_t}\Bigg[&\frac{\partial^2 d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}^2}\left(\sum_{m=1}^{n_x}\frac{\partial g_{j,k}}{\partial x_{m,k}}\frac{\partial x_{m,k}}{\partial q_l}+\frac{\partial g_{j,k}}{\partial q_l}\right)^2+\\ &\frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}}\left(\sum_{m=1}^{n_x}\Big(\big(\sum_{n=1}^{n_x}\frac{\partial^2 g_{j,k}}{\partial x_{n,k}\partial x_{m,k}}\frac{\partial x_{n,k}}{\partial q_l}+\frac{\partial^2 g_{j,k}}{\partial q_l\partial x_{m,k}}\big)\frac{\partial x_{m,k}}{\partial q_l}+\frac{\partial g_{j,k}}{\partial x_{m,k}}\frac{\partial^2 x_{m,k}}{\partial q_l^2}\Big)+\right.\\ &\left.\sum_{m=1}^{n_x}\frac{\partial^2 g_{j,k}}{\partial x_{m,k}\partial q_l}\frac{\partial x_{m,k}}{\partial q_l}+\frac{\partial^2 g_{j,k}}{\partial q_l^2}\right)\Bigg].\end{aligned} \tag{D.51b}$$

Partial derivatives of state values with respect to parameters are generally calculated by numerically solving the sensitivity equations, which are generated by applying the chain rule to the differential equation system.

In the case of an initial value problem, $dx_i/dt = F_i(t, \mathbf{q}, x_1, \ldots, x_{n_x})$ for $i \in \{1, 2, \ldots, n_x\}$, applying the chain rule to $F_i(t, \mathbf{q}, x_1, \ldots, x_{n_x})$ generates the sensitivity equations,

$$\frac{d}{dt}\left(\frac{\partial x_i}{\partial q_l}\right) = \frac{\partial}{\partial q_l}\left(\frac{dx_i}{dt}\right) = \sum_{j=1}^{n_x}\frac{\partial F_i}{\partial x_j}\frac{\partial x_j}{\partial q_l}+\frac{\partial F_i}{\partial q_l}, \tag{D.52a}$$

$$\begin{aligned}&\frac{d}{dt}\left(\frac{\partial^2 x_i}{\partial q_l^2}\right) = \frac{\partial^2}{\partial q_l^2}\left(\frac{dx_i}{dt}\right) =\\ &\sum_{j=1}^{n_x}\left[\left(\sum_{k=1}^{n_x}\frac{\partial^2 F_i}{\partial x_k\partial x_j}\frac{\partial x_k}{\partial q_l}+\frac{\partial^2 F_i}{\partial q_l\partial x_j}\right)\frac{\partial x_j}{\partial q_l}+\frac{\partial F_i}{\partial x_j}\frac{\partial^2 x_j}{\partial q_l^2}\right]+\sum_{j=1}^{n_x}\frac{\partial^2 F_i}{\partial x_j\partial q_l}\frac{\partial x_j}{\partial q_l}+\frac{\partial^2 F_i}{\partial q_l^2} =\\ &\sum_{j=1}^{n_x}\left[\left(\sum_{k=1}^{n_x}\frac{\partial^2 F_i}{\partial x_k\partial x_j}\frac{\partial x_k}{\partial q_l}\right)\frac{\partial x_j}{\partial q_l}+2\frac{\partial^2 F_i}{\partial x_j\partial q_l}\frac{\partial x_j}{\partial q_l}+\frac{\partial F_i}{\partial x_j}\frac{\partial^2 x_j}{\partial q_l^2}\right]+\frac{\partial^2 F_i}{\partial q_l^2},\end{aligned} \tag{D.52b}$$

two systems of differential equations, one in $\partial x_i/\partial q_l$ and one in $\partial^2 x_i/\partial q_l^2$ for $i \in \{1, 2, \ldots, n_x\}$. From Equations (D.52a) and (D.52b), using the forward Euler method, the simplest explicit numerical method for initial value problems, I can calculate $\partial x_{i,m}/\partial q_l$ and $\partial^2 x_{i,m}/\partial q_l^2$ for $i \in \{1, 2, \ldots, n_x\}$ and $m \in \mathcal{I}_\Delta$ such that

$$\frac{\partial x_{i,m+1}}{\partial q_l} = \frac{\partial x_{i,m}}{\partial q_l}+\frac{1}{t_{m+1}-t_m}\left(\sum_{j=1}^{n_x}\frac{\partial F_{i,m}}{\partial x_{j,m}}\frac{\partial x_{j,m}}{\partial q_l}+\frac{\partial F_{i,m}}{\partial q_l}\right), \tag{D.53a}$$

$$\begin{aligned}&\frac{\partial^2 x_{i,m+1}}{\partial q_l^2} = \frac{\partial^2 x_{i,m}}{\partial q_l^2}+\frac{1}{t_{m+1}-t_m}\Bigg(\ \ldots\\ &\sum_{j=1}^{n_x}\left[\left(\sum_{k=1}^{n_x}\frac{\partial^2 F_{i,m}}{\partial x_{k,m}\partial x_{j,m}}\frac{\partial x_{k,m}}{\partial q_l}\right)\frac{\partial x_{j,m}}{\partial q_l}+2\frac{\partial^2 F_{i,m}}{\partial x_{j,m}\partial q_l}\frac{\partial x_{j,m}}{\partial q_l}+\frac{\partial F_{i,m}}{\partial x_{j,m}}\frac{\partial^2 x_{j,m}}{\partial q_l^2}\right]+\frac{\partial^2 F_{i,m}}{\partial q_l^2}\Bigg),\end{aligned} \tag{D.53b}$$

where $F_{i,m} = F_i(t_m, \mathbf{q}, x_{1,m}, \ldots, x_{n_x,m})$.

Solving system (D.53a) for $i \in \{1, 2, \ldots, n_x\}$, $m \in \mathcal{I}_\Delta$, and $l \in \{1, 2, \ldots, n_q\}$ requires calculating partial derivative values,

$$\begin{aligned} &\frac{\partial F_{i,m}}{\partial x_{j,m}} \text{ for all } (i,j,m) \in \{1,\ldots,n_x\} \times \{1,\ldots,n_x\} \times \mathcal{I}_\Delta, \\ &\frac{\partial F_{i,m}}{\partial q_l} \text{ for all } (i,m,l) \in \{1,\ldots,n_x\} \times \mathcal{I}_\Delta \times \{1,\ldots,n_q\}, \end{aligned} \tag{D.54}$$

which, respectively, require

$$\begin{aligned} O(n_{F_1}) \times n_x n_x n_\Delta &= O(n_{F_1} n_x n_x n_\Delta), \\ O(n_{F_1}) \times n_x n_\Delta n_q &= O(n_{F_1} n_x n_\Delta n_q) \end{aligned} \tag{D.55}$$

operations to calculate, as stipulated in equation (D.7). Apart from calculating partial derivative values, solving system (D.53a) for all $i \in \{1, 2, \ldots, n_x\}$, $m \in \mathcal{I}_\Delta$, and $l \in \{1, 2, \ldots, n_q\}$ is equivalent in computational complexity to calculating

$$\begin{aligned} &\frac{\partial F_{i,m}}{\partial x_{j,m}} \frac{\partial x_{j,m}}{\partial q_l} \text{ for all } (i,j,m,l) \in \{1,\ldots,n_x\} \times \{1,\ldots,n_x\} \times \mathcal{I}_\Delta \times \{1,\ldots,n_q\}, \\ &\frac{\partial F_{i,m}}{\partial q_l} \text{ for all } (i,m,l) \in \{1,\ldots,n_x\} \times \mathcal{I}_\Delta \times \{1,\ldots,n_q\}, \end{aligned} \tag{D.56}$$

which, respectively, require

$$\begin{aligned} O(1) \circ O(I_0) \times n_x n_x n_\Delta n_q &= O(n_x n_x n_\Delta n_q), \\ O(I_0) \times n_x n_\Delta n_q &= O(n_x n_\Delta n_q) \end{aligned} \tag{D.57}$$

operations to calculate, where I_0 indicates values that have been calculated previously and $O(I_0) = 1$. Therefore, from complexity counts (D.55) and (D.57), in total, solving systems (D.53a) for all $i \in \{1, 2, \ldots, n_x\}$, $m \in \mathcal{I}_\Delta$, and $l \in \{1, 2, \ldots, n_q\}$, the first order sensitivity equations for an initial value problem using the forward Euler method, requires

$$\begin{aligned} O(n_{F_1} n_x n_x n_\Delta) + O(n_{F_1} n_x n_\Delta n_q) + O(n_x n_x n_\Delta n_q) + O(n_x n_\Delta n_q) = \\ O\big(n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) \end{aligned} \tag{D.58}$$

operations. The forward Euler method applied to an initial value problem is the computationally least expensive explicit numerical solution method. Thus, in general, solving the first order sensitivity equations with an explicit numerical solution method requires

$$O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) \tag{D.59}$$

operations.

Similarly, solving system (D.53b) for $i \in \{1, 2, \ldots, n_x\}$, $m \in \mathcal{I}_\Delta$, and $l \in \{1, 2, \ldots, n_q\}$ requires calculating partial derivative values,

$$\begin{aligned}
&\frac{\partial^2 F_{i,m}}{\partial x_{k,m} \partial x_{j,m}} \text{ for all } (i, j, k, m) \in \{1, \ldots, n_x\} \times \{1, \ldots, n_x\} \times \{1, \ldots, n_x\} \times \mathcal{I}_\Delta, \\
&\frac{\partial^2 F_{i,m}}{\partial x_{j,m} \partial q_l} \text{ for all } (i, j, m, l) \in \{1, \ldots, n_x\} \times \{1, \ldots, n_x\} \times \mathcal{I}_\Delta \times \{1, \ldots, n_q\}, \\
&\frac{\partial^2 F_{i,m}}{\partial q_l^2} \text{ for all } (i, m, l) \in \{1, 2, \ldots, n_x\} \times \mathcal{I}_\Delta \times \{1, 2, \ldots, n_q\},
\end{aligned} \tag{D.60}$$

which, respectively, require

$$\begin{aligned}
O(n_{F_2}) \times n_x n_x n_x n_\Delta &= O(n_{F_2} n_x n_x n_x n_\Delta), \\
O(n_{F_2}) \times n_x n_x n_\Delta n_q &= O(n_{F_2} n_x n_x n_\Delta n_q), \\
O(n_{F_2}) \times n_x n_\Delta n_q &= O(n_{F_2} n_x n_\Delta n_q)
\end{aligned} \tag{D.61}$$

operations to calculate, as stipulated in equation (D.7). Apart from calculating partial derivative values, solving systems (D.53b) for all $i \in \{1, 2, \ldots, n_x\}$, $m \in \mathcal{I}_\Delta$, and $l \in \{1, 2, \ldots, n_q\}$ is equivalent in computational complexity to calculating

$$\begin{aligned}
&\frac{\partial^2 F_{i,m}}{\partial x_{k,m} \partial x_{j,m}} \frac{\partial x_{k,m}}{\partial q_l} \text{ for all } (i, j, k, m, l) \in \{1, \ldots, n_x\}^3 \times \mathcal{I}_\Delta \times \{1, \ldots, n_q\}, \\
&I_0 \cdot \frac{\partial x_{j,m}}{\partial q_l} \text{ for all } (j, m, l) \in \{1, \ldots, n_x\} \times \mathcal{I}_\Delta \times \{1, \ldots, n_q\}, \\
&\frac{\partial^2 F_{i,m}}{\partial x_{j,m} \partial q_l} \frac{\partial x_{j,m}}{\partial q_l} \text{ for all } (i, j, m, l) \in \{1, \ldots, n_x\}^2 \times \mathcal{I}_\Delta \times \{1, \ldots, n_q\}, \\
&\frac{\partial F_{i,m}}{\partial x_{j,m}} \frac{\partial^2 x_{j,m}}{\partial q_l^2} \text{ for all } (i, j, m, l) \in \{1, \ldots, n_x\}^2 \times \mathcal{I}_\Delta \times \{1, \ldots, n_q\}, \\
&\frac{\partial^2 F_{i,m}}{\partial q_l^2} \text{ for all } (i, m, l) \in \{1, \ldots, n_x\} \times \mathcal{I}_\Delta \times \{1, \ldots, n_q\},
\end{aligned} \tag{D.62}$$

which, respectively, require

$$\begin{aligned}
O(1) \circ O(I_0) \times n_x n_x n_x n_\Delta n_q &= O(n_x n_x n_x n_\Delta n_q), \\
O(1) \circ O(I_0) \times n_x n_\Delta n_q &= O(n_x n_\Delta n_q), \\
O(1) \circ O(I_0) \times n_x n_x n_\Delta n_q &= O(n_x n_x n_\Delta n_q), \\
O(1) \circ O(I_0) \times n_x n_x n_\Delta n_q &= O(n_x n_x n_\Delta n_q), \\
O(I_0) \times n_x n_\Delta n_q &= O(n_x n_\Delta n_q)
\end{aligned} \tag{D.63}$$

operations to calculate. Therefore, from complexity counts (D.61) and (D.63), in total, solving systems (D.53b) for all $i \in \{1, 2, \ldots, n_x\}$, $m \in \mathcal{I}_\Delta$, and $l \in \{1, 2, \ldots, n_q\}$, the un-mixed second

order sensitivity equations for an initial value problem using the forward Euler method, requires

$$\begin{gathered}O(n_{F_2}n_xn_xn_xn_\Delta) + O(n_{F_2}n_xn_xn_\Delta n_q) + O(n_{F_2}n_xn_\Delta n_q) + O(n_xn_xn_xn_\Delta n_q) + \\ O(n_xn_\Delta n_q) + O(n_xn_xn_\Delta n_q) + O(n_xn_xn_\Delta n_q) + O(n_xn_\Delta n_q) = \\ O\big(n_xn_xn_\Delta(n_{F_2}n_x + n_{F_2}n_q + n_xn_q)\big)\end{gathered} \tag{D.64}$$

operations. The forward Euler method applied to an initial value problem is the computationally least expensive explicit numerical solution method. Thus, in general, solving the un-mixed second order sensitivity equations with an explicit numerical solution method requires

$$O\big(\geq n_xn_xn_\Delta(n_{F_2}n_x + n_{F_2}n_q + n_xn_q)\big) \tag{D.65}$$

operations.

After solving the first order sensitivity equations, calculating $\partial r(\mathbf{q})/\partial q_l$, of equation (D.51a), for all $l \in \{1, 2, \ldots, n_q\}$ requires calculating partial derivative values,

$$\begin{gathered}\frac{\partial g_{j,k}}{\partial x_{m,k}} \text{ for all } (j,k,m) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_x\}, \\ \frac{\partial g_{j,k}}{\partial q_l} \text{ for all } (j,k,l) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\}, \\ \frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}} \text{ for all } (j,k) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\},\end{gathered} \tag{D.66}$$

which, respectively, require

$$\begin{aligned}O(n_{g_1}) \times n_yn_tn_x &= O(n_{g_1}n_yn_tn_x), \\ O(n_{g_1}) \times n_yn_tn_q &= O(n_{g_1}n_yn_tn_q), \\ O(1) \circ O(I_0) \times n_yn_t &= O(n_yn_t)\end{aligned} \tag{D.67}$$

operations to calculate, as stipulated in Equations (D.3) and (D.4). After solving the first order sensitivity equations and calculating partial derivative values, calculating $\partial r(\mathbf{q})/\partial q_l$, of equation (D.51a), for all $l \in \{1, 2, \ldots, n_q\}$ is equivalent in computational complexity to calculating

$$\begin{gathered}\frac{\partial g_{j,k}}{\partial x_{m,k}}\frac{\partial x_{m,k}}{\partial q_l} \text{ for all } (j,k,l,m) \in \\ \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\} \times \{1,\ldots,n_x\}, \\ \frac{\partial g_{j,k}}{\partial q_l} \text{ for all } (j,k,l) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\}, \\ \frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}} \cdot I_0 \text{ for all } (j,k) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\},\end{gathered} \tag{D.68}$$

which, respectively, require

$$\begin{aligned} O(1) \circ O(I_0) \times n_y n_t n_q n_x &= O(n_y n_t n_q n_x), \\ O(I_0) \times n_y n_t n_q &= O(n_y n_t n_q), \\ O(1) \circ O(I_0) \times n_y n_t &= O(n_y n_t) \end{aligned} \tag{D.69}$$

operations to calculate. Thus, from complexity counts (D.67) and (D.69), after solving the first order sensitivity equations, calculating $\partial r(\mathbf{q})/\partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{aligned} O(n_{g_1} n_y n_t n_x) + O(n_{g_1} n_y n_t n_q) + O(n_y n_t) + O(n_y n_t n_q n_x) + O(n_y n_t n_q) + \\ O(n_y n_t) = O\big(n_y n_t (n_{g_1} n_x + n_{g_1} n_q + n_q n_x)\big) \end{aligned} \tag{D.70}$$

operations. Therefore, from complexity counts (D.59) and (D.70), in total, calculating $\partial r(\mathbf{q})/\partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{aligned} O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + O\big(n_y n_t (n_{g_1} n_x + n_{g_1} n_q + n_q n_x)\big) = \\ O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + O\big(n_y n_t n_{g_1} (n_x + n_q)\big) \end{aligned} \tag{D.71}$$

operations with an explicit numerical solution method, as $n_y \leq n_x$ and $n_t \leq n_\Delta$.

After solving the un-mixed second order sensitivity equations, calculating $\partial^2 r(\mathbf{q})/\partial q_l^2$, of equation (D.51b), for all $l \in \{1, 2, \ldots, n_q\}$ requires calculating partial derivative values,

$$\begin{aligned} \frac{\partial^2 d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}^2} \text{ for all } (j,k) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\}, \\ \frac{\partial^2 g_{j,k}}{\partial x_{n,k} \partial x_{m,k}} \text{ for all } (j,k,m,n) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_x\}^2, \\ \frac{\partial^2 g_{j,k}}{\partial x_{m,k} \partial q_l} \text{ for all } (j,k,l,m) \in \\ \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\} \times \{1,\ldots,n_x\}, \\ \frac{\partial^2 g_{j,k}}{\partial q_l^2} \text{ for all } (j,k,l) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\}, \end{aligned} \tag{D.72}$$

which, respectively, require

$$\begin{aligned} O(1) \circ O(I_0) \times n_y n_t &= O(n_y n_t), \\ O(n_{g_2}) \times n_y n_t n_x n_x &= O(n_{g_2} n_y n_t n_x n_x), \\ O(n_{g_2}) \times n_y n_t n_q n_x &= O(n_{g_2} n_y n_t n_q n_x), \\ O(n_{g_2}) \times n_y n_t n_q &= O(n_{g_2} n_y n_t n_q) \end{aligned} \tag{D.73}$$

operations to calculate, as stipulated in Equations (D.3) and (D.4). After solving the un-

mixed second order sensitivity equations, and calculating partial derivative values, calculating $\partial^2 r(\mathbf{q})/\partial q_l^2$, of equation (D.51b), for all $l \in \{1, 2, \ldots, n_q\}$ is equivalent in computational complexity to calculating

$$\begin{gathered}
\frac{\partial^2 d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}^2} \cdot {I_0}^2 \text{ for all } (j,k) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\}, \\
\frac{\partial^2 g_{j,k}}{\partial x_{n,k} \partial x_{m,k}} \frac{\partial x_{n,k}}{\partial q_l} \text{ for all } (j,k,l,m,n) \in \\
\{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\} \times \{1,\ldots,n_x\}^2, \\
\frac{\partial^2 g_{j,k}}{\partial q_l \partial x_{m,k}} \text{ for all } (j,k,l,m) \in \\
\{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\} \times \{1,\ldots,n_x\}, \\
I_0 \cdot \frac{\partial x_{m,k}}{\partial q_l} \text{ for all } (k,l,m) \in \{1,\ldots,n_t\} \times \{1,\ldots,n_q\} \times \{1,\ldots,n_x\}, \\
\frac{\partial g_{j,k}}{\partial x_{m,k}} \frac{\partial^2 x_{m,k}}{\partial q_l^2} \text{ for all } (j,k,l,m) \in \\
\{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\} \times \{1,\ldots,n_x\}, \\
\frac{\partial^2 g_{j,k}}{\partial x_{m,k} \partial q_l} \frac{\partial x_{m,k}}{\partial q_l} \text{ for all } (j,k,l,m) \in \\
\{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\} \times \{1,\ldots,n_x\}, \\
\frac{\partial^2 g_{j,k}}{\partial q_l^2} \text{ for all } (j,k,l) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\}, \\
\frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}} \cdot I_0 \text{ for all } (j,k) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\},
\end{gathered} \tag{D.74}$$

which, respectively, require

$$\begin{gathered}
O(1) \circ O(I_0) \times n_y n_t = O(n_y n_t), \\
I_{g_2} \cdot O(1) \circ O(I_0) \times n_y n_t n_q n_x n_x = O(I_{g_2} n_y n_t n_q n_x n_x), \\
I_{g_2} \cdot O(I_0) \times n_y n_t n_q n_x = O(I_{g_2} n_y n_t n_q n_x), \\
O(1) \circ O(I_0) \times n_t n_q n_x = O(n_t n_q n_x), \\
O(1) \circ O(I_0) \times n_y n_t n_q n_x = O(n_y n_t n_q n_x), \\
I_{g_2} \cdot O(1) \circ O(I_0) \times n_y n_t n_q n_x = O(I_{g_2} n_y n_t n_q n_x), \\
I_{g_2} \cdot O(I_0) \times n_y n_t n_q = O(I_{g_2} n_y n_t n_q), \\
O(1) \circ O(I_0) \times n_y n_t = O(n_y n_t)
\end{gathered} \tag{D.75}$$

operations to calculate, where

$$I_{g_2} = \begin{cases} 0 & \text{if } n_{g_2} = 0 \\ 1 & \text{if } O(n_{g_2}) \geq 1. \end{cases} \tag{D.76}$$

Thus, from complexity counts (D.73) and (D.75), after solving the un-mixed second order sensitivity equations, calculating $\partial^2 r(\mathbf{q})/\partial q_l^2$ for all $l \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{gathered} O(n_y n_t) + O(n_{g_2} n_y n_t n_x n_x) + O(n_{g_2} n_y n_t n_q n_x) + O(n_{g_2} n_y n_t n_q) + O(n_y n_t) + \\ O(I_{g_2} n_y n_t n_q n_x n_x) + O(I_{g_2} n_y n_t n_q n_x) + O(n_t n_q n_x) + O(n_y n_t n_q n_x) + \\ O(I_{g_2} n_y n_t n_q n_x) + O(I_{g_2} n_y n_t n_q) + O(n_y n_t) = \\ O\big(n_y n_t (n_{g_2} n_x n_x + n_{g_2} n_q n_x + I_{g_2} n_q n_x n_x + n_q n_x)\big) \end{gathered} \tag{D.77}$$

operations. Therefore, from complexity counts (D.65) and (D.77), in total, calculating $\partial^2 r(\mathbf{q})/\partial q_l^2$ for all $l \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{gathered} O\big(\geq n_x n_x n_\Delta (n_{F_2} n_x + n_{F_2} n_q + n_x n_q)\big) + \\ O\big(n_y n_t (n_{g_2} n_x n_x + n_{g_2} n_q n_x + I_{g_2} n_q n_x n_x + n_q n_x)\big) = \\ O\big(\geq n_x n_x n_\Delta (n_{F_2} n_x + n_{F_2} n_q + n_x n_q)\big) + O\big(n_y n_t n_{g_2} (n_x n_x + n_q n_x)\big) \end{gathered} \tag{D.78}$$

operations with an explicit numerical solution method, as $n_y \leq n_x$ and $n_t \leq n_\Delta$.

In each iteration of $r(\mathbf{q})$ descent, I generate $O(n_\sigma)$ line-search test points, one for each value of σ_j. Generating each test point requires an update of all parameter values, requiring $O(n_q)$ operations. From complexity count (D.50), for an explicit numerical solution method, calculating $r(\mathbf{q})$ at each line-search test point requires $O(n_f n_x n_\Delta + n_g n_y n_t)$ operations. Thus, for an explicit numerical solution method, updating all parameter values and calculating $r(\mathbf{q})$ at each line-search test point requires

$$O(n_q) + O(n_f n_x n_\Delta + n_g n_y n_t) = O(n_f n_x n_\Delta + n_g n_y n_t) \tag{D.79}$$

operations, as I consider $r(\mathbf{q})$ with fewer parameter values than state values, $n_q < n_x n_\Delta$. As such, for an explicit numerical solution method, updating all parameter values and calculating $r(\mathbf{q})$ at all line-search test points, calculating first order partial derivatives of $r(\mathbf{q})$ with respect to all parameters, and calculating un-mixed second order partial derivatives of $r(\mathbf{q})$ with respect

to all parameters requires

$$\begin{gathered} O\big(n_\sigma(n_f n_x n_\Delta + n_g n_y n_t)\big) + \\ O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + O\big(n_y n_t n_{g_1}(n_x + n_q)\big) + \\ O\big(\geq n_x n_x n_\Delta (n_{F_2} n_x + n_{F_2} n_q + n_x n_q)\big) + O\big(n_y n_t n_{g_2}(n_x n_x + n_q n_x)\big) = \\ O\big(n_\sigma(n_f n_x n_\Delta + n_g n_y n_t) + n_y n_t n_{g_1}(n_x + n_q)\big) + O\big(n_y n_t n_{g_2}(n_x n_x + n_q n_x)\big) + \\ O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_x n_x n_q)\big) \end{gathered} \tag{D.80}$$

operations, with partial derivative complexity counts from (D.71) and (D.78). Therefore, for an explicit numerical solution method, an iteration of $r(\mathbf{q})$ descent requires

$$\begin{gathered} O\big(n_\sigma(n_f n_x n_\Delta + n_g n_y n_t) + n_y n_t n_{g_1}(n_x + n_q)\big) + O\big(n_y n_t n_{g_2}(n_x n_x + n_q n_x)\big) + \\ O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_x n_x n_q)\big) \end{gathered} \tag{D.81}$$

operations.

For implicit numerical solution methods, $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ is an implicit system of equations in $\mathbf{x}$ and discretized sensitivity equations are implicit systems of equations in discretized sensitivity values, $\partial x_{i,m}/\partial q_l$ and $\partial^2 x_{i,m}/\partial q_l^2$. Otherwise, $r(\mathbf{q})$ descent with an implicit numerical solution method is identical to $r(\mathbf{q})$ descent with an explicit numerical solution method. Generally, $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ is a nonlinear system of equations that is solved numerically using Newton's method. Each iteration of Newton's method to solve $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ requires calculating the values of $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x})$ and the values of first order partial derivatives of $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x})$ with respect to $x_{l,m}$, for all $i \in \{1, 2, \ldots, n_x\}$, $k \in \mathcal{I}_\Delta$, $l \in \{1, 2, \ldots, n_x\}$, and $m \in \mathcal{I}_\Delta$. Calculating

$$f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) \text{ for all } (i, k) \in \{1, 2, \ldots, n_x\} \times \mathcal{I}_\Delta \tag{D.82}$$

requires

$$O(n_f) \times n_x n_\Delta = O(n_f n_x n_\Delta) \tag{D.83}$$

operations, as stipulated in equation (D.5). $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x})$ depend on $x_{l,m}$ for only a small fraction of k in $\mathcal{I}_\Delta$, at k in $\mathcal{I}_{\Delta_m} \subset \mathcal{I}_\Delta$. Calculating

$$\frac{\partial f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x})}{\partial x_{l,m}} \text{ for all } (i, k, l, m) \in \{1, \ldots, n_x\} \times \mathcal{I}_{\Delta_m} \times \{1, \ldots, n_x\} \times \mathcal{I}_\Delta \tag{D.84}$$

requires

$$O(n_{f_1}) \times n_x n_\delta n_x n_\Delta = O(n_{f_1} n_x n_\delta n_x n_\Delta) \tag{D.85}$$

operations, as stipulated in equation (D.5). Additionally, each iteration of Newton's method to

solve $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ requires solving matrix equations, requiring a total of $O(n_M)$ operations. Thus, in conjunction with complexity counts (D.83) and (D.85), each iteration of Newton's method to solve $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ requires

$$\begin{aligned} O(n_x n_\Delta n_f) + O(n_{f_1} n_x n_\delta n_x n_\Delta) + O(n_M) = \\ O(n_x n_\Delta n_f + n_{f_1} n_x n_\delta n_x n_\Delta + n_M) \end{aligned} \tag{D.86}$$

operations. Solving $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ requires $O(n_N)$ iterations of Newton's method. Thus, for an implicit numerical solution method, numerically solving $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ to determine $\mathbf{x}$ requires

$$O(n_N n_x n_\Delta n_f + n_N n_{f_1} n_x n_\delta n_x n_\Delta + n_N n_M) \tag{D.87}$$

operations. From complexity count (D.49), After calculating $\mathbf{x}$, calculating $r(\mathbf{q})$ requires calculating

$$\frac{1}{n_y} \sum_{j=1}^{n_y} \sum_{k=1}^{n_t} d_{\mathrm{y}_{j,k}}\big(g_{j,k}(\mathbf{q}, \mathbf{x})\big),$$

which requires $O(n_g n_y n_t)$ operations to calculate. Thus, from complexity counts (D.49) and (D.87), calculating $r(\mathbf{q})$ requires

$$\begin{aligned} O(n_N n_x n_\Delta n_f + n_N n_{f_1} n_x n_\delta n_x n_\Delta + n_N n_M) + O(n_g n_y n_t) = \\ O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big) \end{aligned} \tag{D.88}$$

operations with an implicit numerical solution method.

The sensitivity equations are generated by applying the chain rule to the differential equation system, and are thus linear in sensitivity values, $\partial x_i / \partial q_l$ and $\partial^2 x_i / \partial q_l^2$. As such, discretized sensitivity equations are generally linear in discretized sensitivity values, $\partial x_{i,m} / \partial q_l$ and $\partial^2 x_{i,m} / \partial q_l^2$. Thus, beyond the calculations required to solve the discretized sensitivity equations with an explicit numerical solution method, solving the discretized sensitivity equations with an implicit numerical solution method requires solving matrix equations. Both first order and un-mixed second order discretized sensitivity equations with respect to q_l are identical in size to $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$. Thus, calculating matrix equations in solving first order discretized sensitivity equations with respect to q_l requires a total of $O(n_M)$ operations, and calculating matrix equations in solving un-mixed second order discretized sensitivity equations with respect to q_l requires a total of $O(n_M)$ operations. As such, calculating matrix equations in solving first order discretized sensitivity equations with respect to q_l for all $l \in \{1, 2, \ldots, n_q\}$ requires a total of $O(n_q n_M)$ operations, and calculating matrix equations in solving un-mixed second order discretized sensitivity equations with respect to q_l for all $l \in \{1, 2, \ldots, n_q\}$ requires a total of $O(n_q n_M)$ operations. Therefore, in general, in conjunction with complexity count (D.59), solving

the first order sensitivity equations with an implicit numerical solution method requires

$$O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + O(n_q n_M) \tag{D.89}$$

operations, and, in conjunction with complexity count (D.65), solving the un-mixed second order sensitivity equations with an implicit numerical solution method requires

$$O\big(\geq n_x n_x n_\Delta (n_{F_2} n_x + n_{F_2} n_q + n_x n_q)\big) + O(n_q n_M) \tag{D.90}$$

operations. From complexity count (D.70), after solving the first order sensitivity equations, calculating $\partial r(\mathbf{q})/\partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$ requires $O(n_y n_t (n_{g_1} n_x + n_{g_1} n_q + n_q n_x))$ operations. Therefore, from complexity counts (D.89) and (D.70), in total, calculating $\partial r(\mathbf{q})/\partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{gathered} O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + O(n_q n_M) + \\ O\big(n_y n_t (n_{g_1} n_x + n_{g_1} n_q + n_q n_x)\big) = \\ O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + O\big(n_q n_M + n_y n_t n_{g_1} (n_x + n_q)\big) \end{gathered} \tag{D.91}$$

operations with an implicit numerical solution method, as $n_y \leq n_x$ and $n_t \leq n_\Delta$. From complexity count (D.77), after solving the un-mixed second order sensitivity equations, calculating $\partial^2 r(\mathbf{q})/\partial q_l^2$ for all $l \in \{1, 2, \ldots, n_q\}$ requires $O(n_y n_t (n_{g_2} n_x n_x + n_{g_2} n_q n_x + I_{g_2} n_q n_x n_x + n_q n_x))$ operations. Therefore, from complexity counts (D.90) and (D.77), in total, calculating $\partial^2 r(\mathbf{q})/\partial q_l^2$ for all $l \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{gathered} O\big(\geq n_x n_x n_\Delta (n_{F_2} n_x + n_{F_2} n_q + n_x n_q)\big) + O(n_q n_M) \\ O\big(n_y n_t (n_{g_2} n_x n_x + n_{g_2} n_q n_x + I_{g_2} n_q n_x n_x + n_q n_x)\big) = \\ O\big(\geq n_x n_x n_\Delta (n_{F_2} n_x + n_{F_2} n_q + n_x n_q)\big) + O(n_q n_M) + O\big(n_y n_t n_{g_2} (n_x n_x + n_q n_x)\big) \end{gathered} \tag{D.92}$$

operations with an implicit numerical solution method, as $n_y \leq n_x$ and $n_t \leq n_\Delta$.

In each iteration of $r(\mathbf{q})$ descent, I generate $O(n_\sigma)$ line-search test points, one for each value of σ_j. Generating each test point requires an update of all parameter values, requiring $O(n_q)$ operations. From complexity count (D.88), for an implicit numerical solution method, calculating $r(\mathbf{q})$ at each line-search test point requires $O(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t)$ operations. Thus, for an implicit numerical solution method, updating all parameter values and calculating $r(\mathbf{q})$ at each line-search test point requires

$$\begin{gathered} O(n_q) + O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big) = \\ O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big) \end{gathered} \tag{D.93}$$

operations, as I consider $r(\mathbf{q})$ with fewer parameter values than state values, $n_q < n_x n_\Delta$. As

such, for an implicit numerical solution method, updating all parameter values and calculating $r(\mathbf{q})$ at all line-search test points, calculating first order partial derivatives of $r(\mathbf{q})$ with respect to all parameters, and calculating un-mixed second order partial derivatives of $r(\mathbf{q})$ with respect to all parameters requires

$$\begin{gathered}
O\big(n_\sigma n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_\sigma n_N n_M + n_\sigma n_g n_y n_t\big) + \\
O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + O\big(n_q n_M + n_y n_t n_{g_1} (n_x + n_q)\big) + \\
O\big(\geq n_x n_x n_\Delta (n_{F_2} n_x + n_{F_2} n_q + n_x n_q)\big) + O(n_q n_M) + O\big(n_y n_t n_{g_2} (n_x n_x + n_q n_x)\big) \\
= O\big(n_\sigma n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_\sigma n_g n_y n_t\big) + \\
O\big(n_M (n_q + n_\sigma n_N) + n_y n_t n_{g_1} (n_x + n_q)\big) + O\big(n_y n_t n_{g_2} (n_x n_x + n_q n_x)\big) + \\
O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_x n_x n_q)\big)
\end{gathered} \tag{D.94}$$

operations, with partial derivative complexity counts from complexity counts (D.91) and (D.92). Therefore, for an implicit numerical solution method, an iteration of $r(\mathbf{q})$ descent requires

$$\begin{gathered}
O\big(n_\sigma n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_\sigma n_g n_y n_t\big) + \\
O\big(n_M (n_q + n_\sigma n_N) + n_y n_t n_{g_1} (n_x + n_q)\big) + O\big(n_y n_t n_{g_2} (n_x n_x + n_q n_x)\big) + \\
O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_x n_x n_q)\big)
\end{gathered} \tag{D.95}$$

operations.

Alternatively, I can approximate partial derivatives of $r(\mathbf{q})$ with respect to parameters by finite differences, rather than by solving the sensitivity equations. Most simply,

$$\frac{\partial r(\mathbf{q})}{\partial q_l} \approx \frac{r(\mathbf{q} + h_l \mathbf{e}_l) - r(\mathbf{q})}{h_l}, \tag{D.96a}$$

$$\frac{\partial^2 r(\mathbf{q})}{\partial q_l^2} \approx \frac{r(\mathbf{q} + h_l \mathbf{e}_l) - 2r(\mathbf{q}) + r(\mathbf{q} - h_l \mathbf{e}_l)}{h_l^2}, \tag{D.96b}$$

where $\mathbf{e}_l$ is the l^{th} standard basis vector and h_l is some small perturbation in parameter q_l. Approximating $\partial r(\mathbf{q})/\partial q_l$ and $\partial^2 r(\mathbf{q})/\partial q_l^2$, as in Equations (D.96a) and (D.96b), for all $l \in \{1, 2, \ldots, n_q\}$ is equivalent in computational complexity to calculating

$$\begin{gathered}
r(\mathbf{q} + h_l \mathbf{e}_l) \text{ for all } l \in \{1, 2, \ldots, n_q\}, \\
r(\mathbf{q}), \\
r(\mathbf{q} - h_l \mathbf{e}_l) \text{ for all } l \in \{1, 2, \ldots, n_q\},
\end{gathered} \tag{D.97}$$

which, from complexity count (D.50), for an explicit numerical solution method, respectively,

require

$$\begin{gathered} O(n_f n_x n_\Delta + n_g n_y n_t) \times n_q = O\big(n_q(n_f n_x n_\Delta + n_g n_y n_t)\big), \\ O(I_0), \\ O(n_f n_x n_\Delta + n_g n_y n_t) \times n_q = O\big(n_q(n_f n_x n_\Delta + n_g n_y n_t)\big), \end{gathered} \tag{D.98}$$

operations to calculate, and, from complexity count (D.88), for an implicit numerical solution method, respectively, require

$$\begin{gathered} O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big) \times n_q = \\ O\big(n_N n_x n_\Delta n_q (n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q + n_g n_y n_t n_q\big), \\ O(I_0), \\ O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big) \times n_q = \\ O\big(n_N n_x n_\Delta n_q (n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q + n_g n_y n_t n_q\big) \end{gathered} \tag{D.99}$$

operations to calculate. Thus, for an explicit numerical solution method, approximating $\partial r(\mathbf{q})/\partial q_l$ and $\partial^2 r(\mathbf{q})/\partial q_l^2$ by finite difference for all $l \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{gathered} O\big(n_q(n_f n_x n_\Delta + n_g n_y n_t)\big) + O(I_0) + O\big(n_q(n_f n_x n_\Delta + n_g n_y n_t)\big) = \\ O\big(n_q(n_f n_x n_\Delta + n_g n_y n_t)\big) \end{gathered} \tag{D.100}$$

operations, and for an implicit numerical solution method, approximating $\partial r(\mathbf{q})/\partial q_l$ and $\partial^2 r(\mathbf{q})/\partial q_l^2$ by finite difference for all $l \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{gathered} O\big(n_N n_x n_\Delta n_q (n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q + n_g n_y n_t n_q\big) + O(I_0) + \\ O\big(n_N n_x n_\Delta n_q (n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q + n_g n_y n_t n_q\big) = \\ O\big(n_N n_x n_\Delta n_q (n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q + n_g n_y n_t n_q\big) \end{gathered} \tag{D.101}$$

operations.

For an explicit numerical solution method, updating all parameter values and calculating $r(\mathbf{q})$ at all of the $O(n_\sigma)$ line-search test points, as calculated for each line-search test point in complexity count (D.79), and approximating first order and un-mixed second order partial derivatives of $r(\mathbf{q})$ with respect to all parameters by finite difference, as calculated in complexity count (D.100), requires

$$\begin{gathered} O(n_\sigma n_f n_x n_\Delta + n_\sigma n_g n_y n_t) + O\big(n_q(n_f n_x n_\Delta + n_g n_y n_t)\big) = \\ O\big(n_f n_x n_\Delta (n_\sigma + n_q) + n_g n_y n_t (n_\sigma + n_q)\big) = \\ O\big((n_\sigma + n_q)(n_f n_x n_\Delta + n_g n_y n_t)\big) \end{gathered} \tag{D.102}$$

operations. For an implicit numerical solution method, updating all parameter values and calculating $r(\mathbf{q})$ at all of the $O(n_\sigma)$ line-search test points, as calculated for each line-search test point in complexity count (D.93), and approximating first order and un-mixed second order partial derivatives of $r(\mathbf{q})$ with respect to all parameters by finite difference, as calculated in complexity count (D.101), requires

$$\begin{gathered}
O\big(n_\sigma n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_\sigma n_N n_M + n_\sigma n_g n_y n_t\big) + \\
O\big(n_N n_x n_\Delta n_q (n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q + n_g n_y n_t n_q\big) = \\
O\big(n_N n_x n_\Delta (n_\sigma + n_q)(n_f + n_{f_1} n_x n_\delta) + n_N n_M (n_\sigma + n_q) + n_g n_y n_t (n_\sigma + n_q)\big) = \\
O\Big(\big(n_\sigma + n_q\big)\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big)\Big)
\end{gathered} \tag{D.103}$$

operations. Therefore, from complexity count (D.102), for an explicit numerical solution method, an iteration of $r(\mathbf{q})$ descent with partial derivative approximation by finite difference requires

$$O\big((n_\sigma + n_q)(n_f n_x n_\Delta + n_g n_y n_t)\big) \tag{D.104}$$

operations, and from complexity count (D.103), for an implicit numerical solution method, an iteration of $r(\mathbf{q})$ descent with partial derivative approximation by finite difference requires

$$O\Big(\big(n_\sigma + n_q\big)\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big)\Big) \tag{D.105}$$

operations. □

D.2.2 Counting the Computational Complexity of Newton's Method to Minimize $r(\mathbf{q})$

On an unrestricted domain, a local minimum of $r(\mathbf{q})$ occurs where $\partial r(\mathbf{q})/\partial p_l = 0$ for all $l \in \{1, 2, \ldots, n_q\}$. Thus, a local minimum of $r(\mathbf{q})$ is calculable by applying Newton's method to find a solution to the system $\partial r(\mathbf{q})/\partial p_l = 0$ for all $l \in \{1, 2, \ldots, n_q\}$.

Theorem 9. *For an explicit numerical solution method, an iteration of $r(\mathbf{q})$ minimization by Newton's method requires*

$$\begin{gathered}
O\big(n_f n_x n_\Delta + n_y n_t n_{g_1}(n_x + n_q)\big) + O\big(n_y n_t n_{g_2}(n_x n_x + n_q n_x + n_q n_q)\big) + \\
O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_{F_2} n_q n_q + n_x n_x n_q + n_x n_q n_q)\big)
\end{gathered} \tag{D.106}$$

operations. For an implicit numerical solution method, an iteration of $r(\mathbf{q})$ minimization by

Newton's method requires

$$\begin{gathered} O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_M (n_q n_q + n_N) + n_y n_t n_{g_1} (n_x + n_q)\big) + \\ O\big(n_y n_t n_{g_2} (n_x n_x + n_q n_x + n_q n_q)\big) + \\ O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_{F_2} n_q n_q + n_x n_x n_q + n_x n_q n_q)\big) \end{gathered} \tag{D.107}$$

operations, with $O(n_M)$ operations in solving matrix equations for each of $O(n_N)$ iterations of Newton's method applied to $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$, for all $i \in \{1, 2, \ldots, n_x\}$ and $k \in \mathcal{I}_\Delta$.

Alternatively, for an explicit numerical solution method, an iteration of $r(\mathbf{q})$ minimization by Newton's method with partial derivative approximation by finite difference requires

$$O\big(n_q n_q (n_f n_x n_\Delta + n_g n_y n_t)\big) \tag{D.108}$$

operations, and, for an implicit numerical solution method, an iteration of $r(\mathbf{q})$ minimization by Newton's method with partial derivative approximation by finite difference requires

$$O\big(n_N n_x n_\Delta n_q n_q (n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q n_q + n_g n_y n_t n_q n_q\big) \tag{D.109}$$

operations.

Proof. In each iteration of $r(\mathbf{q})$ minimization by Newton's method, I calculate numerical solution values, $\mathbf{x}$. For an explicit numerical solution method, $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ is an explicit system of equations in $\mathbf{x}$, and solving $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ simply requires evaluating $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x})$. Thus, to determine $\mathbf{x}$, solving

$$f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0 \text{ for all } (i, k) \in \{1, \ldots, n_x\} \times \mathcal{I}_\Delta \tag{D.110}$$

requires

$$O(n_f) \times n_x n_\Delta = O(n_f n_x n_\Delta) \tag{D.111}$$

operations, as stipulated in equation (D.5).

In each iteration of $r(\mathbf{q})$ minimization by Newton's method, I calculate first order and second

order partial derivatives of $r(\mathbf{q})$ with respect to all parameters,

$$\frac{\partial r(\mathbf{q})}{\partial q_l} = \frac{1}{n_y}\sum_{j=1}^{n_y}\sum_{k=1}^{n_t}\frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}}\left(\sum_{m=1}^{n_x}\frac{\partial g_{j,k}}{\partial x_{m,k}}\frac{\partial x_{m,k}}{\partial q_l}+\frac{\partial g_{j,k}}{\partial q_l}\right), \tag{D.112a}$$

$$\begin{aligned}\frac{\partial^2 r(\mathbf{q})}{\partial q_m \partial q_l} = \frac{1}{n_y}\sum_{j=1}^{n_y}\sum_{k=1}^{n_t}\Bigg[&\frac{\partial^2 d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}^2}\left(\sum_{s=1}^{n_x}\frac{\partial g_{j,k}}{\partial x_{s,k}}\frac{\partial x_{s,k}}{\partial q_m}+\frac{\partial g_{j,k}}{\partial q_m}\right)\left(\sum_{s=1}^{n_x}\frac{\partial g_{j,k}}{\partial x_{s,k}}\frac{\partial x_{s,k}}{\partial q_l}+\frac{\partial g_{j,k}}{\partial q_l}\right)+\\ &\frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}}\left(\sum_{s=1}^{n_x}\Big(\big(\sum_{t=1}^{n_x}\frac{\partial^2 g_{j,k}}{\partial x_{t,k}\partial x_{s,k}}\frac{\partial x_{t,k}}{\partial q_m}+\frac{\partial^2 g_{j,k}}{\partial q_m \partial x_{s,k}}\big)\frac{\partial x_{s,k}}{\partial q_l}+\frac{\partial g_{j,k}}{\partial x_{s,k}}\frac{\partial^2 x_{s,k}}{\partial q_m \partial q_l}\Big)+\right.\\ &\left.\sum_{s=1}^{n_x}\frac{\partial^2 g_{j,k}}{\partial x_{s,k}\partial q_l}\frac{\partial x_{s,k}}{\partial q_m}+\frac{\partial^2 g_{j,k}}{\partial q_m \partial q_l}\right)\Bigg].\end{aligned} \tag{D.112b}$$

Partial derivatives of state values with respect to parameters are generally calculated by numerically solving the sensitivity equations, which are generated by applying the chain rule to the differential equation system.

In the case of an initial value problem, $dx_i/dt = F_i(t, \mathbf{q}, x_1, \ldots, x_{n_x})$ for $i \in \{1, 2, \ldots, n_x\}$, applying the chain rule to $F_i(t, \mathbf{q}, x_1, \ldots, x_{n_x})$ generates the sensitivity equations,

$$\frac{d}{dt}\left(\frac{\partial x_i}{\partial q_l}\right) = \frac{\partial}{\partial q_l}\left(\frac{dx_i}{dt}\right) = \sum_{j=1}^{n_x}\frac{\partial F_i}{\partial x_j}\frac{\partial x_j}{\partial q_l}+\frac{\partial F_i}{\partial q_l}, \tag{D.113a}$$

$$\begin{aligned}&\frac{d}{dt}\left(\frac{\partial^2 x_i}{\partial q_m \partial q_l}\right) = \frac{\partial^2}{\partial q_m \partial q_l}\left(\frac{dx_i}{dt}\right) =\\ &\sum_{j=1}^{n_x}\left[\left(\sum_{k=1}^{n_x}\frac{\partial^2 F_i}{\partial x_k \partial x_j}\frac{\partial x_k}{\partial q_m}+\frac{\partial^2 F_i}{\partial q_m \partial x_j}\right)\frac{\partial x_j}{\partial q_l}+\frac{\partial F_i}{\partial x_j}\frac{\partial^2 x_j}{\partial q_m \partial q_l}\right]+\\ &\sum_{j=1}^{n_x}\frac{\partial^2 F_i}{\partial x_j \partial q_l}\frac{\partial x_j}{\partial q_m}+\frac{\partial^2 F_i}{\partial q_m \partial q_l},\end{aligned} \tag{D.113b}$$

two systems of differential equations, one in $\partial x_i/\partial q_l$ and one in $\partial^2 x_i/\partial q_m \partial q_l$ for $i \in \{1, 2, \ldots, n_x\}$. From Equations (D.113a) and (D.113b), using the forward Euler method, the simplest explicit numerical method for initial value problems, I can calculate $\partial x_{i,s}/\partial q_l$ and $\partial^2 x_{i,s}/\partial q_m \partial q_l$ for

$i \in \{1, 2, \ldots, n_x\}$ and $s \in \mathcal{I}_\Delta$ such that

$$\frac{\partial x_{i,s+1}}{\partial q_l} = \frac{\partial x_{i,s}}{\partial q_l} + \frac{1}{t_{s+1} - t_s} \left(\sum_{j=1}^{n_x} \frac{\partial F_{i,s}}{\partial x_{j,s}} \frac{\partial x_{j,s}}{\partial q_l} + \frac{\partial F_{i,s}}{\partial q_l} \right), \tag{D.114a}$$

$$\begin{aligned} \frac{\partial^2 x_{i,s+1}}{\partial q_m \partial q_l} = \frac{\partial^2 x_{i,s}}{\partial q_m \partial q_l} + \frac{1}{t_{s+1} - t_s} \Bigg(\ldots \\ \sum_{j=1}^{n_x} \left[\left(\sum_{k=1}^{n_x} \frac{\partial^2 F_{i,s}}{\partial x_{k,s} \partial x_{j,s}} \frac{\partial x_{k,s}}{\partial q_m} + \frac{\partial^2 F_{i,s}}{\partial q_m \partial x_{j,s}} \right) \frac{\partial x_{j,s}}{\partial q_l} + \frac{\partial F_{i,s}}{\partial x_{j,s}} \frac{\partial^2 x_{j,s}}{\partial q_m \partial q_l} \right] + \\ \sum_{j=1}^{n_x} \frac{\partial^2 F_{i,s}}{\partial x_{j,s} \partial q_l} \frac{\partial x_{j,s}}{\partial q_m} + \frac{\partial^2 F_{i,s}}{\partial q_m \partial q_l} \Bigg), \end{aligned} \tag{D.114b}$$

where $F_{i,s} = F_i(t_s, \mathbf{q}, x_{1,s}, \ldots, x_{n_x,s})$.

Solving system (D.114a) for $i \in \{1, 2, \ldots, n_x\}$, $l \in \{1, 2, \ldots, n_q\}$, and $s \in \mathcal{I}_\Delta$ requires calculating partial derivative values,

$$\begin{aligned} &\frac{\partial F_{i,s}}{\partial x_{j,s}} \text{ for all } (i, j, s) \in \{1, \ldots, n_x\} \times \{1, \ldots, n_x\} \times \mathcal{I}_\Delta, \\ &\frac{\partial F_{i,s}}{\partial q_l} \text{ for all } (i, l, s) \in \{1, \ldots, n_x\} \times \{1, \ldots, n_q\} \times \mathcal{I}_\Delta, \end{aligned} \tag{D.115}$$

which, respectively, require

$$\begin{aligned} O(n_{F_1}) \times n_x n_x n_\Delta = O(n_{F_1} n_x n_x n_\Delta), \\ O(n_{F_1}) \times n_x n_q n_\Delta = O(n_{F_1} n_x n_q n_\Delta) \end{aligned} \tag{D.116}$$

operations to calculate, as stipulated in equation (D.7). Apart from calculating partial derivative values, solving system (D.114a) for all $i \in \{1, 2, \ldots, n_x\}$, $l \in \{1, 2, \ldots, n_q\}$, and $s \in \mathcal{I}_\Delta$ is equivalent in computational complexity to calculating

$$\begin{aligned} &\frac{\partial F_{i,s}}{\partial x_{j,s}} \frac{\partial x_{j,s}}{\partial q_l} \text{ for all } (i, j, l, s) \in \{1, \ldots, n_x\} \times \{1, \ldots, n_x\} \times \{1, \ldots, n_q\} \times \mathcal{I}_\Delta, \\ &\frac{\partial F_{i,s}}{\partial q_l} \text{ for all } (i, l, s) \in \{1, \ldots, n_x\} \times \{1, \ldots, n_q\} \times \mathcal{I}_\Delta, \end{aligned} \tag{D.117}$$

which, respectively, require

$$\begin{aligned} O(1) \circ O(I_0) \times n_x n_x n_q n_\Delta = O(n_x n_x n_q n_\Delta), \\ O(I_0) \times n_x n_q n_\Delta = O(n_x n_q n_\Delta) \end{aligned} \tag{D.118}$$

operations to calculate, where I_0 indicates values that have been calculated previously and $O(I_0) = 1$. Therefore, from complexity counts (D.116) and (D.118), in total, solving systems (D.114a) for all $i \in \{1, 2, \ldots, n_x\}$, $l \in \{1, 2, \ldots, n_q\}$, and $s \in \mathcal{I}_\Delta$, the first order sensitivity

equations for an initial value problem using the forward Euler method, requires

$$O(n_{F_1} n_x n_x n_\Delta) + O(n_{F_1} n_x n_q n_\Delta) + O(n_x n_x n_q n_\Delta) + O(n_x n_q n_\Delta) = O\big(n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) \tag{D.119}$$

operations. The forward Euler method applied to an initial value problem is the computationally least expensive explicit numerical solution method. Thus, in general, solving the first order sensitivity equations with an explicit numerical solution method requires

$$O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) \tag{D.120}$$

operations.

Similarly, solving system (D.114b) for $i \in \{1, 2, \ldots, n_x\}$, $l \in \{1, 2, \ldots, n_q\}$, $m \in \{1, 2, \ldots, n_q\}$, and $s \in \mathcal{I}_\Delta$ requires calculating partial derivative values,

$$\begin{aligned} &\frac{\partial^2 F_{i,s}}{\partial x_{k,s} \partial x_{j,s}} \text{ for all } (i,j,k,s) \in \{1,\ldots,n_x\} \times \{1,\ldots,n_x\} \times \{1,\ldots,n_x\} \times \mathcal{I}_\Delta, \\ &\frac{\partial^2 F_{i,s}}{\partial x_{j,s} \partial q_l} \text{ for all } (i,j,l,s) \in \{1,\ldots,n_x\} \times \{1,\ldots,n_x\} \times \{1,\ldots,n_q\} \times \mathcal{I}_\Delta, \\ &\frac{\partial^2 F_{i,s}}{\partial q_m \partial q_l} \text{ for all } (i,l,m,s) \in \{1,2,\ldots,n_x\} \times \{1,2,\ldots,n_q\} \times \{1,2,\ldots,n_q\} \times \mathcal{I}_\Delta, \end{aligned} \tag{D.121}$$

which, respectively, require

$$\begin{aligned} O(n_{F_2}) \times n_x n_x n_x n_\Delta &= O(n_{F_2} n_x n_x n_x n_\Delta), \\ O(n_{F_2}) \times n_x n_x n_q n_\Delta &= O(n_{F_2} n_x n_x n_q n_\Delta), \\ O(n_{F_2}) \times n_x n_q n_q n_\Delta &= O(n_{F_2} n_x n_q n_q n_\Delta) \end{aligned} \tag{D.122}$$

operations to calculate, as stipulated in equation (D.7). Apart from calculating partial derivative values, solving systems (D.114b) for all $i \in \{1, 2, \ldots, n_x\}$, $l \in \{1, 2, \ldots, n_q\}$, $m \in \{1, 2, \ldots, n_q\}$,

and $s \in \mathcal{I}_\Delta$ is equivalent in computational complexity to calculating

$$\begin{gathered}
\frac{\partial^2 F_{i,s}}{\partial x_{k,s} \partial x_{j,s}} \frac{\partial x_{k,s}}{\partial q_m} \text{ for all } (i,j,k,m,s) \in \{1,\ldots,n_x\}^3 \times \{1,\ldots,n_q\} \times \mathcal{I}_\Delta, \\
\frac{\partial^2 F_{i,s}}{\partial q_m \partial x_{j,s}} \text{ for all } (i,j,m,s) \in \{1,\ldots,n_x\}^2 \times \{1,\ldots,n_q\} \times \mathcal{I}_\Delta, \\
I_0 \cdot \frac{\partial x_{j,s}}{\partial q_l} \text{ for all } (j,l,s) \in \{1,\ldots,n_x\} \times \{1,\ldots,n_q\} \times \mathcal{I}_\Delta, \\
\frac{\partial F_{i,s}}{\partial x_{j,s}} \frac{\partial^2 x_{j,s}}{\partial q_m \partial q_l} \text{ for all } (i,j,l,m,s) \in \{1,\ldots,n_x\}^2 \times \{1,\ldots,n_q\}^2 \times \mathcal{I}_\Delta, \\
\frac{\partial^2 F_{i,s}}{\partial x_{j,s} \partial q_l} \frac{\partial x_{j,s}}{\partial q_m} \text{ for all } (i,j,l,m,s) \in \{1,\ldots,n_x\}^2 \times \{1,\ldots,n_q\}^2 \times \mathcal{I}_\Delta, \\
\frac{\partial^2 F_{i,s}}{\partial q_m \partial q_l} \text{ for all } (i,l,m,s) \in \{1,\ldots,n_x\} \times \{1,\ldots,n_q\}^2 \times \mathcal{I}_\Delta,
\end{gathered} \tag{D.123}$$

which, respectively, require

$$\begin{gathered}
O(1) \circ O(I_0) \times n_x n_x n_x n_q n_\Delta = O(n_x n_x n_x n_q n_\Delta), \\
O(I_0) \times n_x n_x n_q n_\Delta = O(n_x n_x n_q n_\Delta), \\
O(1) \circ O(I_0) \times n_x n_q n_\Delta = O(n_x n_q n_\Delta), \\
O(1) \circ O(I_0) \times n_x n_x n_q n_q n_\Delta = O(n_x n_x n_q n_q n_\Delta), \\
O(1) \circ O(I_0) \times n_x n_x n_q n_q n_\Delta = O(n_x n_x n_q n_q n_\Delta), \\
O(I_0) \times n_x n_q n_q n_\Delta = O(n_x n_q n_q n_\Delta)
\end{gathered} \tag{D.124}$$

operations to calculate. Therefore, from complexity counts (D.122) and (D.124), in total, solving systems (D.114b) for all $i \in \{1,2,\ldots,n_x\}$, $l \in \{1,2,\ldots,n_q\}$, $m \in \{1,2,\ldots,n_q\}$, and $s \in \mathcal{I}_\Delta$, the second order sensitivity equations for an initial value problem using the forward Euler method, requires

$$\begin{gathered}
O(n_{F_2} n_x n_x n_x n_\Delta) + O(n_{F_2} n_x n_x n_q n_\Delta) + O(n_{F_2} n_x n_q n_q n_\Delta) + \\
O(n_x n_x n_x n_q n_\Delta) + O(n_x n_x n_q n_\Delta) + O(n_x n_q n_\Delta) + \\
O(n_x n_x n_q n_q n_\Delta) + O(n_x n_x n_q n_q n_\Delta) + O(n_x n_q n_q n_\Delta) = \\
O\big(n_x n_\Delta (n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_{F_2} n_q n_q + n_x n_x n_q + n_x n_q n_q)\big)
\end{gathered} \tag{D.125}$$

operations. The forward Euler method applied to an initial value problem is the computationally least expensive explicit numerical solution method. Thus, in general, solving the second order sensitivity equations with an explicit numerical solution method requires

$$O\big(\geq n_x n_\Delta (n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_{F_2} n_q n_q + n_x n_x n_q + n_x n_q n_q)\big) \tag{D.126}$$

operations.

After solving the first order sensitivity equations, calculating $\partial r(\mathbf{q})/\partial q_l$, of equation (D.112a), for all $l \in \{1, 2, \ldots, n_q\}$ requires calculating partial derivative values,

$$\begin{aligned} &\frac{\partial g_{j,k}}{\partial x_{m,k}} \text{ for all } (j,k,m) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_x\}, \\ &\frac{\partial g_{j,k}}{\partial q_l} \text{ for all } (j,k,l) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\}, \\ &\frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}} \text{ for all } (j,k) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\}, \end{aligned} \tag{D.127}$$

which, respectively, require

$$\begin{aligned} O(n_{g_1}) \times n_y n_t n_x &= O(n_{g_1} n_y n_t n_x), \\ O(n_{g_1}) \times n_y n_t n_q &= O(n_{g_1} n_y n_t n_q), \\ O(1) \circ O(I_0) \times n_y n_t &= O(n_y n_t) \end{aligned} \tag{D.128}$$

operations to calculate, as stipulated in Equations (D.3) and (D.4). After solving the first order sensitivity equations and calculating partial derivative values, calculating $\partial r(\mathbf{q})/\partial q_l$, of equation (D.112a), for all $l \in \{1, 2, \ldots, n_q\}$ is equivalent in computational complexity to calculating

$$\begin{aligned} &\frac{\partial g_{j,k}}{\partial x_{m,k}} \frac{\partial x_{m,k}}{\partial q_l} \text{ for all } (j,k,m,l) \in \\ &\{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_x\} \times \{1,\ldots,n_q\}, \\ &\frac{\partial g_{j,k}}{\partial q_l} \text{ for all } (j,k,l) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\}, \\ &\frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}} \cdot I_0 \text{ for all } (j,k) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\}, \end{aligned} \tag{D.129}$$

which, respectively, require

$$\begin{aligned} O(1) \circ O(I_0) \times n_y n_t n_x n_q &= O(n_y n_t n_x n_q), \\ O(I_0) \times n_y n_t n_q &= O(n_y n_t n_q), \\ O(1) \circ O(I_0) \times n_y n_t &= O(n_y n_t) \end{aligned} \tag{D.130}$$

operations to calculate. Thus, from complexity counts (D.128) and (D.130), after solving the first order sensitivity equations, calculating $\partial r(\mathbf{q})/\partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{aligned} &O(n_{g_1} n_y n_t n_x) + O(n_{g_1} n_y n_t n_q) + O(n_y n_t) + O(n_y n_t n_x n_q) + O(n_y n_t n_q) + \\ &O(n_y n_t) = O\big(n_y n_t (n_{g_1} n_x + n_{g_1} n_q + n_x n_q)\big) \end{aligned} \tag{D.131}$$

operations. Therefore, from complexity counts (D.120) and (D.131), in total, calculating

$\partial r(\mathbf{q})/\partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{aligned} &O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + O\big(n_y n_t (n_{g_1} n_x + n_{g_1} n_q + n_x n_q)\big) = \\ &\quad O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + O\big(n_y n_t n_{g_1} (n_x + n_q)\big) \end{aligned} \tag{D.132}$$

operations with an explicit numerical solution method, as $n_y \leq n_x$ and $n_t \leq n_\Delta$.

After solving the second order sensitivity equations, calculating $\partial^2 r(\mathbf{q})/\partial q_m \partial q_l$, of equation (D.112b), for all $l \in \{1, 2, \ldots, n_q\}$ and $m \in \{1, 2, \ldots, n_q\}$ requires calculating partial derivative values,

$$\begin{aligned} \frac{\partial^2 d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}^2} &\text{ for all } (j,k) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\}, \\ \frac{\partial^2 g_{j,k}}{\partial x_{t,k} \partial x_{s,k}} &\text{ for all } (j,k,s,t) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_x\}^2, \\ \frac{\partial^2 g_{j,k}}{\partial x_{s,k} \partial q_l} &\text{ for all } (j,k,l,s) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\} \times \{1,\ldots,n_x\}, \\ \frac{\partial^2 g_{j,k}}{\partial q_m \partial q_l} &\text{ for all } (j,k,l,m) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\}^2, \end{aligned} \tag{D.133}$$

which, respectively, require

$$\begin{aligned} O(1) \circ O(I_0) \times n_y n_t &= O(n_y n_t), \\ O(n_{g_2}) \times n_y n_t n_x n_x &= O(n_{g_2} n_y n_t n_x n_x), \\ O(n_{g_2}) \times n_y n_t n_q n_x &= O(n_{g_2} n_y n_t n_q n_x), \\ O(n_{g_2}) \times n_y n_t n_q n_q &= O(n_{g_2} n_y n_t n_q n_q) \end{aligned} \tag{D.134}$$

operations to calculate, as stipulated in Equations (D.3) and (D.4). After solving the second order sensitivity equations and calculating partial derivative values, calculating $\partial^2 r(\mathbf{q})/\partial q_m \partial q_l$, of equation (D.112b), for all $l \in \{1, 2, \ldots, n_q\}$ and $m \in \{1, 2, \ldots, n_q\}$ is equivalent in computational

complexity to calculating

$$\begin{gathered}
\frac{\partial^2 d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}^2} \cdot I_0 \text{ for all } (j,k) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\}, \\
\frac{\partial^2 g_{j,k}}{\partial x_{t,k} \partial x_{s,k}} \frac{\partial x_{t,k}}{\partial q_m} \text{ for all } (j,k,m,s,t) \in \\
\{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\} \times \{1,\ldots,n_x\}^2, \\
\frac{\partial^2 g_{j,k}}{\partial q_m \partial x_{s,k}} \text{ for all } (j,k,m,s) \in \\
\{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\} \times \{1,\ldots,n_x\}, \\
I_0 \cdot \frac{\partial x_{s,k}}{\partial q_l} \text{ for all } (k,l,s) \in \{1,\ldots,n_t\} \times \{1,\ldots,n_q\} \times \{1,\ldots,n_x\}, \\
\frac{\partial g_{j,k}}{\partial x_{s,k}} \frac{\partial^2 x_{s,k}}{\partial q_m \partial q_l} \text{ for all } (j,k,l,m,s) \in \\
\{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\}^2 \times \{1,\ldots,n_x\}, \\
\frac{\partial^2 g_{j,k}}{\partial x_{s,k} \partial q_l} \frac{\partial x_{s,k}}{\partial q_m} \text{ for all } (j,k,l,m,s) \in \\
\{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\}^2 \times \{1,\ldots,n_x\}, \\
\frac{\partial^2 g_{j,k}}{\partial q_m \partial q_l} \text{ for all } (j,k,l,m) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\}^2, \\
\frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}} \cdot I_0 \text{ for all } (j,k) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\},
\end{gathered} \tag{D.135}$$

which, respectively, require

$$\begin{gathered}
O(1) \circ O(I_0) \times n_y n_t = O(n_y n_t), \\
I_{g_2} \cdot O(1) \circ O(I_0) \times n_y n_t n_q n_x n_x = O(I_{g_2} n_y n_t n_q n_x n_x), \\
I_{g_2} \cdot O(I_0) \times n_y n_t n_q n_x = O(I_{g_2} n_y n_t n_q n_x), \\
O(1) \circ O(I_0) \times n_t n_q n_x = O(n_t n_q n_x), \\
O(1) \circ O(I_0) \times n_y n_t n_q n_q n_x = O(n_y n_t n_q n_q n_x), \\
I_{g_2} \cdot O(1) \circ O(I_0) \times n_y n_t n_q n_q n_x = O(I_{g_2} n_y n_t n_q n_q n_x), \\
I_{g_2} \cdot O(I_0) \times n_y n_t n_q n_q = O(I_{g_2} n_y n_t n_q n_q), \\
O(1) \circ O(I_0) \times n_y n_t = O(n_y n_t)
\end{gathered} \tag{D.136}$$

operations to calculate, where

$$I_{g_2} = \begin{cases} 0 & \text{if } n_{g_2} = 0 \\ 1 & \text{if } O(n_{g_2}) \geq 1. \end{cases} \tag{D.137}$$

Thus, from complexity counts (D.134) and (D.136), after solving the second order sensitivity

equations, calculating $\partial^2 r(\mathbf{q})/\partial q_m \partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$ and $m \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{gathered}
O(n_y n_t) + O(n_{g_2} n_y n_t n_x n_x) + O(n_{g_2} n_y n_t n_q n_x) + O(n_{g_2} n_y n_t n_q n_q) + O(n_y n_t) + \\
O(I_{g_2} n_y n_t n_q n_x n_x) + O(I_{g_2} n_y n_t n_q n_x) + O(n_t n_q n_x) + O(n_y n_t n_q n_q n_x) + \\
O(I_{g_2} n_y n_t n_q n_q n_x) + O(I_{g_2} n_y n_t n_q n_q) + O(n_y n_t) = \\
O\big(n_y n_t (n_{g_2} n_x n_x + n_{g_2} n_q n_x + n_{g_2} n_q n_q + I_{g_2} n_q n_x n_x + I_{g_2} n_q n_q n_x + n_q n_q n_x)\big)
\end{gathered} \tag{D.138}$$

operations. Therefore, from complexity counts (D.126) and (D.138), in total, calculating $\partial^2 r(\mathbf{q})/\partial q_m \partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$ and $m \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{gathered}
O\big(\geq n_x n_\Delta (n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_{F_2} n_q n_q + n_x n_x n_q + n_x n_q n_q)\big) + \\
O\big(n_y n_t (n_{g_2} n_x n_x + n_{g_2} n_q n_x + n_{g_2} n_q n_q + I_{g_2} n_q n_x n_x + I_{g_2} n_q n_q n_x + n_q n_q n_x)\big) = \\
O\big(\geq n_x n_\Delta (n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_{F_2} n_q n_q + n_x n_x n_q + n_x n_q n_q)\big) + \\
O\big(n_y n_t n_{g_2} (n_x n_x + n_q n_x + n_q n_q)\big)
\end{gathered} \tag{D.139}$$

operations with an explicit numerical solution method, as $n_y \leq n_x$ and $n_t \leq n_\Delta$.

In each iteration of $r(\mathbf{q})$ minimization by Newton's method, updating parameter values requires solving an $n_q \times n_q$ matrix equation, which requires $O(n_q^3)$ operations using Gaussian elimination. Thus, for an explicit numerical solution method, calculating first order partial derivatives of $r(\mathbf{q})$ with respect to all parameters, calculating second order partial derivatives of $r(\mathbf{q})$ with respect to all parameters, updating parameter values, and updating numerical solution values requires

$$\begin{gathered}
O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + O\big(n_y n_t n_{g_1} (n_x + n_q)\big) + \\
O\big(\geq n_x n_\Delta (n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_{F_2} n_q n_q + n_x n_x n_q + n_x n_q n_q)\big) + \\
O\big(n_y n_t n_{g_2} (n_x n_x + n_q n_x + n_q n_q)\big) + O(n_q^3) + O(n_f n_x n_\Delta) = \\
O\big(n_f n_x n_\Delta + n_y n_t n_{g_1} (n_x + n_q)\big) + O\big(n_y n_t n_{g_2} (n_x n_x + n_q n_x + n_q n_q)\big) \\
O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_{F_2} n_q n_q + n_x n_x n_q + n_x n_q n_q)\big)
\end{gathered} \tag{D.140}$$

operations, as $n_q < n_x n_\Delta$, the number of parameters is less than the number of state values, with partial derivative complexity counts from (D.132) and (D.139) and numerical solution complexity count from (D.111). Therefore, for an explicit numerical solution method, an iteration of $r(\mathbf{q})$ minimization by Newton's method requires

$$\begin{gathered}
O\big(n_f n_x n_\Delta + n_y n_t n_{g_1} (n_x + n_q)\big) + O\big(n_y n_t n_{g_2} (n_x n_x + n_q n_x + n_q n_q)\big) \\
O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_{F_2} n_q n_q + n_x n_x n_q + n_x n_q n_q)\big)
\end{gathered} \tag{D.141}$$

operations.

For implicit numerical solution methods, $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ is an implicit system of equations in

$\mathbf{x}$ and discretized sensitivity equations are implicit systems of equations in discretized sensitivity values, $\partial x_{i,s}/\partial q_l$ and $\partial^2 x_{i,s}/\partial q_m \partial q_l$. Otherwise, $r(\mathbf{q})$ minimization by Newton's method with an implicit numerical solution method is identical to $r(\mathbf{q})$ minimization by Newton's method with an explicit numerical solution method. Generally, $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ is a nonlinear system of equations that is solved numerically using Newton's method. Each iteration of Newton's method to solve $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ requires calculating the values of $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x})$ and the values of first order partial derivatives of $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x})$ with respect to $x_{l,m}$, for all $i \in \{1, 2, \ldots, n_x\}$, $k \in \mathcal{I}_\Delta$, $l \in \{1, 2, \ldots, n_x\}$, and $m \in \mathcal{I}_\Delta$. Calculating

$$f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) \text{ for all } (i,k) \in \{1, 2, \ldots, n_x\} \times \mathcal{I}_\Delta \tag{D.142}$$

requires

$$O(n_f) \times n_x n_\Delta = O(n_f n_x n_\Delta) \tag{D.143}$$

operations, as stipulated in equation (D.5). $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x})$ depend on $x_{l,m}$ for only a small fraction of k in $\mathcal{I}_\Delta$, at k in $\mathcal{I}_{\Delta_m} \subset \mathcal{I}_\Delta$. Calculating

$$\frac{\partial f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x})}{\partial x_{l,m}} \text{ for all } (i,k,l,m) \in \{1, \ldots, n_x\} \times \mathcal{I}_{\Delta_m} \times \{1, \ldots, n_x\} \times \mathcal{I}_\Delta \tag{D.144}$$

requires

$$O(n_{f_1}) \times n_x n_\delta n_x n_\Delta = O(n_{f_1} n_x n_\delta n_x n_\Delta) \tag{D.145}$$

operations, as stipulated in equation (D.5). Additionally, each iteration of Newton's method to solve $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ requires solving matrix equations, requiring a total of $O(n_M)$ operations. Thus, in conjunction with complexity counts (D.143) and (D.145), each iteration of Newton's method to solve $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ requires

$$\begin{gathered} O(n_x n_\Delta n_f) + O(n_{f_1} n_x n_\delta n_x n_\Delta) + O(n_M) = \\ O\big(n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_M\big) \end{gathered} \tag{D.146}$$

operations. Solving $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ requires $O(n_N)$ iterations of Newton's method. Thus, for an implicit numerical solution method, numerically solving $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ to determine $\mathbf{x}$ requires

$$O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M\big) \tag{D.147}$$

operations.

The sensitivity equations are generated by applying the chain rule to the differential equation system, and are thus linear in sensitivity values, $\partial x_i/\partial q_l$ and $\partial^2 x_i/\partial q_m \partial q_l$. As such, discretized sensitivity equations are generally linear in discretized sensitivity values, $\partial x_{i,s}/\partial q_l$

and $\partial^2 x_{i,s}/\partial q_m \partial q_l$. Thus, beyond the calculations required to solve the discretized sensitivity equations with an explicit numerical solution method, solving the discretized sensitivity equations with an implicit numerical solution method requires solving matrix equations. Both first order and second order discretized sensitivity equations with respect to q_l are identical in size to $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$. Thus, calculating matrix equations in solving first order discretized sensitivity equations with respect to q_l requires a total of $O(n_M)$ operations, and calculating matrix equations in solving second order discretized sensitivity equations with respect to q_l requires a total of $O(n_M)$ operations. As such, calculating matrix equations in solving first order discretized sensitivity equations with respect to q_l for all $l \in \{1, 2, \ldots, n_q\}$ requires a total of $O(n_q n_M)$ operations, and calculating matrix equations in solving second order discretized sensitivity equations with respect to q_l and q_m for all $l \in \{1, 2, \ldots, n_q\}$ and $m \in \{1, 2, \ldots, n_q\}$ requires a total of $O(n_q n_q n_M)$ operations. Therefore, in general, in conjunction with complexity count (D.120), solving the first order sensitivity equations with an implicit numerical solution method requires

$$O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + O(n_q n_M) \tag{D.148}$$

operations, and, in conjunction with complexity count (D.126), solving the second order sensitivity equations with an implicit numerical solution method requires

$$O\big(\geq n_x n_\Delta (n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_{F_2} n_q n_q + n_x n_x n_q + n_x n_q n_q)\big) + O(n_q n_q n_M) \tag{D.149}$$

operations. From complexity count (D.131), after solving the first order sensitivity equations, calculating $\partial r(\mathbf{q})/\partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$ requires $O(n_y n_t (n_{g_1} n_x + n_{g_1} n_q + n_x n_q))$ operations. Therefore, from complexity counts (D.148) and (D.131), in total, calculating $\partial r(\mathbf{q})/\partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{gathered} O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + O(n_q n_M) + \\ O\big(n_y n_t (n_{g_1} n_x + n_{g_1} n_q + n_x n_q)\big) = \\ O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + O\big(n_q n_M + n_y n_t n_{g_1} (n_x + n_q)\big) \end{gathered} \tag{D.150}$$

operations with an implicit numerical solution method, as $n_y \leq n_x$ and $n_t \leq n_\Delta$. From complexity count (D.138), after solving the second order sensitivity equations, calculating $\partial^2 r(\mathbf{q})/\partial q_m \partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$ and $m \in \{1, 2, \ldots, n_q\}$ requires $O(n_y n_t (n_{g_2} n_x n_x + n_{g_2} n_q n_x + n_{g_2} n_q n_q + I_{g_2} n_q n_x n_x + I_{g_2} n_q n_q n_x + n_q n_q n_x))$ operations. Therefore, from complexity counts (D.149) and (D.138), in total, calculating $\partial^2 r(\mathbf{q})/\partial q_m \partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$ and $m \in \{1, 2, \ldots, n_q\}$

requires

$$\begin{gathered}O\big(\geq n_x n_\Delta (n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_{F_2} n_q n_q + n_x n_x n_q + n_x n_q n_q)\big) + O(n_q n_q n_M) + \\ O\big(n_y n_t (n_{g_2} n_x n_x + n_{g_2} n_q n_x + n_{g_2} n_q n_q + I_{g_2} n_q n_x n_x + I_{g_2} n_q n_q n_x + n_q n_q n_x)\big) = \\ O\big(\geq n_x n_\Delta (n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_{F_2} n_q n_q + n_x n_x n_q + n_x n_q n_q)\big) + O(n_q n_q n_M) + \\ O\big(n_y n_t n_{g_2} (n_x n_x + n_q n_x + n_q n_q)\big)\end{gathered} \tag{D.151}$$

operations with an implicit numerical solution method, as $n_y \leq n_x$ and $n_t \leq n_\Delta$.

In each iteration of $r(\mathbf{q})$ minimization by Newton's method, updating parameter values requires solving an $n_q \times n_q$ matrix equation, which requires $O(n_q^3)$ operations using Gaussian elimination. Thus, for an implicit numerical solution method, calculating first order partial derivatives of $r(\mathbf{q})$ with respect to all parameters, calculating second order partial derivatives of $r(\mathbf{q})$ with respect to all parameters, updating parameter values, and updating numerical solution values requires

$$\begin{gathered}O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + O\big(n_q n_M + n_y n_t n_{g_1} (n_x + n_q)\big) + \\ O\big(\geq n_x n_\Delta (n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_{F_2} n_q n_q + n_x n_x n_q + n_x n_q n_q)\big) + O(n_q n_q n_M) + \\ O\big(n_y n_t n_{g_2} (n_x n_x + n_q n_x + n_q n_q)\big) + O(n_q^3) + \\ O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M\big) = \\ O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_M (n_q n_q + n_N) + n_y n_t n_{g_1} (n_x + n_q)\big) + \\ O\big(n_y n_t n_{g_2} (n_x n_x + n_q n_x + n_q n_q)\big) + \\ O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_{F_2} n_q n_q + n_x n_x n_q + n_x n_q n_q)\big)\end{gathered} \tag{D.152}$$

operations, as $n_q < n_x n_\Delta$, the number of parameters is less than the number of state values, with partial derivative complexity counts from (D.150) and (D.151) and numerical solution complexity count from (D.147). Therefore, for an implicit numerical solution method, an iteration of $r(\mathbf{q})$ minimization by Newton's method requires

$$\begin{gathered}O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_M (n_q n_q + n_N) + n_y n_t n_{g_1} (n_x + n_q)\big) + \\ O\big(n_y n_t n_{g_2} (n_x n_x + n_q n_x + n_q n_q)\big) + \\ O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_{F_2} n_x n_x + n_{F_2} n_x n_q + n_{F_2} n_q n_q + n_x n_x n_q + n_x n_q n_q)\big)\end{gathered} \tag{D.153}$$

operations.

Alternatively, I can approximate partial derivatives of $r(\mathbf{q})$ with respect to parameters by

finite differences, rather than by solving the sensitivity equations. Most simply,

$$\frac{\partial r(\mathbf{q})}{\partial q_l} \approx \frac{r(\mathbf{q}+h_l\mathbf{e}_l)-r(\mathbf{q})}{h_l}, \tag{D.154a}$$

$$\frac{\partial^2 r(\mathbf{q})}{\partial q_m \partial q_l} \approx \frac{r(\mathbf{q}+h_l\mathbf{e}_l)-r(\mathbf{q})-r(\mathbf{q}+h_l\mathbf{e}_l-h_m\mathbf{e}_m)+r(\mathbf{q}-h_m\mathbf{e}_m)}{h_m h_l}, \tag{D.154b}$$

where $\mathbf{e}_l$ is the l^{th} standard basis vector and h_l is some small perturbation in parameter q_l.

After calculating $\mathbf{x}$, calculating $r(\mathbf{q})$ requires calculating

$$\frac{1}{n_y}\sum_{j=1}^{n_y}\sum_{k=1}^{n_t} d_{\mathrm{y}_{j,k}}\big(g_{j,k}(\mathbf{q},\mathbf{x})\big), \tag{D.155}$$

which is equivalent in computational complexity to calculating

$$d_{\mathrm{y}_{j,k}}\big(g_{j,k}(\mathbf{q},\mathbf{x})\big) \text{ for all } (j,k) \in \{1,\dots,n_y\}\times\{1,\dots,n_t\}, \tag{D.156}$$

which requires

$$O(1)\circ O(n_g)\times n_y n_t = O(n_g n_y n_t) \tag{D.157}$$

operations to calculate, as stipulated in Equations (D.3) and (D.4). Thus, from complexity counts (D.111) and (D.157), calculating $r(\mathbf{q})$ requires

$$O(n_f n_x n_\Delta) + O(n_g n_y n_t) = O(n_f n_x n_\Delta + n_g n_y n_t) \tag{D.158}$$

operations with an explicit numerical solution method, and, from complexity counts (D.147) and (D.157), calculating $r(\mathbf{q})$ requires

$$\begin{gathered} O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M\big) + O(n_g n_y n_t) = \\ O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big) \end{gathered} \tag{D.159}$$

operations with an implicit numerical solution method.

Approximating $\partial r(\mathbf{q})/\partial q_l$ and $\partial^2 r(\mathbf{q})/\partial q_m \partial q_l$, as in Equations (D.154a) and (D.154b), for all $l \in \{1,2,\dots,n_q\}$ and $m \in \{1,2,\dots,n_q\}$ is equivalent in computational complexity to calculating

$$\begin{gathered} r(\mathbf{q}+h_l\mathbf{e}_l) \text{ for all } l \in \{1,2,\dots,n_q\}, \\ r(\mathbf{q}), \\ r(\mathbf{q}+h_l\mathbf{e}_l-h_m\mathbf{e}_m) \text{ for all } (l,m) \in \{1,2,\dots,n_q\}^2 \setminus \{(l,m) : l = m\} \\ r(\mathbf{q}-h_m\mathbf{e}_m) \text{ for all } m \in \{1,2,\dots,n_q\}, \end{gathered} \tag{D.160}$$

which, from complexity count (D.158), for an explicit numerical solution method, respectively,

require

$$\begin{gathered}
O(n_f n_x n_\Delta + n_g n_y n_t) \times n_q = O\big(n_q(n_f n_x n_\Delta + n_g n_y n_t)\big), \\
O(n_f n_x n_\Delta + n_g n_y n_t), \\
O(n_f n_x n_\Delta + n_g n_y n_t) \times (n_q n_q - n_q) = O\big(n_q n_q(n_f n_x n_\Delta + n_g n_y n_t)\big), \\
O(n_f n_x n_\Delta + n_g n_y n_t) \times n_q = O\big(n_q(n_f n_x n_\Delta + n_g n_y n_t)\big),
\end{gathered} \tag{D.161}$$

operations to calculate, and, from complexity count (D.159), for an implicit numerical solution method, respectively, require

$$\begin{gathered}
O\big(n_N n_x n_\Delta(n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big) \times n_q = \\
O\big(n_N n_x n_\Delta n_q(n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q + n_g n_y n_t n_q\big), \\
O\big(n_N n_x n_\Delta(n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big), \\
O\big(n_N n_x n_\Delta(n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big) \times (n_q n_q - n_q) = \\
O\big(n_N n_x n_\Delta n_q n_q(n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q n_q + n_g n_y n_t n_q n_q\big), \\
O\big(n_N n_x n_\Delta(n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big) \times n_q = \\
O\big(n_N n_x n_\Delta n_q(n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q + n_g n_y n_t n_q\big)
\end{gathered} \tag{D.162}$$

operations to calculate. Thus, for an explicit numerical solution method, approximating $\partial r(\mathbf{q})/\partial q_l$ and $\partial^2 r(\mathbf{q})/\partial q_m \partial q_l$ by finite difference for all $l \in \{1, 2, \ldots, n_q\}$ and $m \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{gathered}
O\big(n_q(n_f n_x n_\Delta + n_g n_y n_t)\big) + O(n_f n_x n_\Delta + n_g n_y n_t) + \\
O\big(n_q n_q(n_f n_x n_\Delta + n_g n_y n_t)\big) + O\big(n_q(n_f n_x n_\Delta + n_g n_y n_t)\big) = \\
O\big(n_q n_q(n_f n_x n_\Delta + n_g n_y n_t)\big)
\end{gathered} \tag{D.163}$$

operations, and for an implicit numerical solution method, approximating $\partial r(\mathbf{q})/\partial q_l$ and $\partial^2 r(\mathbf{q})/\partial q_m \partial q_l$ by finite difference for all $l \in \{1, 2, \ldots, n_q\}$ and $m \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{gathered}
O\big(n_N n_x n_\Delta n_q(n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q + n_g n_y n_t n_q\big) + \\
O\big(n_N n_x n_\Delta(n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big) + \\
O\big(n_N n_x n_\Delta n_q n_q(n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q n_q + n_g n_y n_t n_q n_q\big) + \\
O\big(n_N n_x n_\Delta n_q(n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q + n_g n_y n_t n_q\big) = \\
O\big(n_N n_x n_\Delta n_q n_q(n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q n_q + n_g n_y n_t n_q n_q\big)
\end{gathered} \tag{D.164}$$

operations.

In each iteration of $r(\mathbf{q})$ minimization by Newton's method, updating parameter values requires solving an $n_q \times n_q$ matrix equation, which requires $O(n_q^3)$ operations using Gaussian

elimination. Thus, for an explicit numerical solution method, approximating first order and second order partial derivatives of $r(\mathbf{q})$ with respect to all parameters by finite difference, updating parameter values, and updating numerical solution values requires

$$O\big(n_q n_q (n_f n_x n_\Delta + n_g n_y n_t)\big) + O(n_q^3) + O(n_f n_x n_\Delta) = O\big(n_q n_q (n_f n_x n_\Delta + n_g n_y n_t)\big) \tag{D.165}$$

operations, as $n_q < n_x n_\Delta$, the number of parameters is less than the number of state values, with partial derivative approximation complexity count from (D.163) and numerical solution complexity count from (D.111); and for an implicit numerical solution method, approximating first order and second order partial derivatives of $r(\mathbf{q})$ with respect to all parameters by finite difference, updating parameter values, and updating numerical solution values requires

$$\begin{gathered} O\big(n_N n_x n_\Delta n_q n_q (n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q n_q + n_g n_y n_t n_q n_q\big) + O(n_q^3) + \\ O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M\big) = \\ O\big(n_N n_x n_\Delta n_q n_q (n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q n_q + n_g n_y n_t n_q n_q\big) \end{gathered} \tag{D.166}$$

operations, as $n_q < n_x n_\Delta$, the number of parameters is less than the number of state values, with partial derivative approximation complexity count from (D.164) and numerical solution complexity count from (D.147). Therefore, from complexity count (D.165), for an explicit numerical solution method, an iteration of $r(\mathbf{q})$ minimization by Newton's method with partial derivative approximation by finite difference requires

$$O\big(n_q n_q (n_f n_x n_\Delta + n_g n_y n_t)\big) \tag{D.167}$$

operations, and from complexity count (D.166), for an implicit numerical solution method, an iteration of $r(\mathbf{q})$ minimization by Newton's method with partial derivative approximation by finite difference requires

$$O\big(n_N n_x n_\Delta n_q n_q (n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q n_q + n_g n_y n_t n_q n_q\big) \tag{D.168}$$

operations. □

D.2.3 Counting the Computational Complexity of Gradient-Based Methods to Minimize $r(\mathbf{q})$

In methods such as the Gauss-Newton method, Levenberg-Marquardt method, and quasi-Newton methods, rather than generating the Hessian matrix by calculating second order partial derivative values, $\partial^2 r(\mathbf{q})/\partial q_m \partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$ and $m \in \{1, 2, \ldots, n_q\}$, as in Newton's method, the Hessian matrix is approximated using first order partial derivative values, $\partial r(\mathbf{q})/\partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$. Thus, an iteration of a method that approximates the Hessian using $\partial r(\mathbf{q})/\partial q_l$

for all $l \in \{1, 2, \ldots, n_q\}$ requires at least as many operations as the number of operations required to calculate $\partial r(\mathbf{q})/\partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$.

Theorem 10. *For an explicit numerical solution method, calculating first order partial derivatives of $r(\mathbf{q})$ with respect to all parameters requires*

$$O\big(n_f n_x n_\Delta + n_y n_t n_{g_1}(n_x + n_q)\big) + O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) \tag{D.169}$$

operations, and, for an implicit numerical solution method, calculating first order partial derivatives of $r(\mathbf{q})$ with respect to all parameters requires

$$\begin{aligned} O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_M (n_q + n_N) + n_y n_t n_{g_1}(n_x + n_q)\big) + \\ O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) \end{aligned} \tag{D.170}$$

operations, with $O(n_M)$ operations in solving matrix equations for each of $O(n_N)$ iterations of Newton's method applied to $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$, for all $i \in \{1, 2, \ldots, n_x\}$ and $k \in \mathcal{I}_\Delta$.

Alternatively, for an explicit numerical solution method, approximating first order partial derivatives of $r(\mathbf{q})$ with respect to all parameters by finite difference requires

$$O\big(n_q (n_f n_x n_\Delta + n_g n_y n_t)\big) \tag{D.171}$$

operations, and, for an implicit numerical solution method, approximating first order partial derivatives of $r(\mathbf{q})$ with respect to all parameters by finite difference requires

$$O\big(n_N n_x n_\Delta n_q (n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q + n_g n_y n_t n_q\big) \tag{D.172}$$

operations.

Proof. For an explicit numerical solution method, $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ is an explicit system of equations in $\mathbf{x}$, and solving $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ simply requires evaluating $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x})$. Thus, to determine $\mathbf{x}$, solving

$$f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0 \text{ for all } (i, k) \in \{1, \ldots, n_x\} \times \mathcal{I}_\Delta \tag{D.173}$$

requires

$$O(n_f) \times n_x n_\Delta = O(n_f n_x n_\Delta) \tag{D.174}$$

operations, as stipulated in equation (D.5).

I calculate first order partial derivatives of $r(\mathbf{q})$ with respect to all parameters,

$$\frac{\partial r(\mathbf{q})}{\partial q_l} = \frac{1}{n_y} \sum_{j=1}^{n_y} \sum_{k=1}^{n_t} \frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}} \left(\sum_{m=1}^{n_x} \frac{\partial g_{j,k}}{\partial x_{m,k}} \frac{\partial x_{m,k}}{\partial q_l} + \frac{\partial g_{j,k}}{\partial q_l} \right). \tag{D.175}$$

Partial derivatives of state values with respect to parameters are generally calculated by numerically solving the sensitivity equations, which are generated by applying the chain rule to the differential equation system.

In the case of an initial value problem, $dx_i/dt = F_i(t, \mathbf{q}, x_1, \ldots, x_{n_x})$ for $i \in \{1, 2, \ldots, n_x\}$, applying the chain rule to $F_i(t, \mathbf{q}, x_1, \ldots, x_{n_x})$ generates the sensitivity equations,

$$\frac{d}{dt}\left(\frac{\partial x_i}{\partial q_l}\right) = \frac{\partial}{\partial q_l}\left(\frac{dx_i}{dt}\right) = \sum_{j=1}^{n_x} \frac{\partial F_i}{\partial x_j}\frac{\partial x_j}{\partial q_l} + \frac{\partial F_i}{\partial q_l}, \tag{D.176}$$

a system of differential equations in $\partial x_i/\partial q_l$ for $i \in \{1, 2, \ldots, n_x\}$. From Equations (D.176), using the forward Euler method, the simplest explicit numerical method for initial value problems, I can calculate $\partial x_{i,m}/\partial q_l$ for $i \in \{1, 2, \ldots, n_x\}$ and $m \in \mathcal{I}_\Delta$ such that

$$\frac{\partial x_{i,m+1}}{\partial q_l} = \frac{\partial x_{i,m}}{\partial q_l} + \frac{1}{t_{m+1} - t_m}\left(\sum_{j=1}^{n_x} \frac{\partial F_{i,m}}{\partial x_{j,m}}\frac{\partial x_{j,m}}{\partial q_l} + \frac{\partial F_{i,m}}{\partial q_l}\right), \tag{D.177}$$

where $F_{i,m} = F_i(t_m, \mathbf{q}, x_{1,m}, \ldots, x_{n_x,m})$. Solving system (D.177) for $i \in \{1, 2, \ldots, n_x\}$, $l \in \{1, 2, \ldots, n_q\}$, and $m \in \mathcal{I}_\Delta$ requires calculating partial derivative values,

$$\begin{aligned} &\frac{\partial F_{i,m}}{\partial x_{j,m}} \text{ for all } (i, j, m) \in \{1, \ldots, n_x\} \times \{1, \ldots, n_x\} \times \mathcal{I}_\Delta, \\ &\frac{\partial F_{i,m}}{\partial q_l} \text{ for all } (i, l, m) \in \{1, \ldots, n_x\} \times \{1, \ldots, n_q\} \times \mathcal{I}_\Delta, \end{aligned} \tag{D.178}$$

which, respectively, require

$$\begin{aligned} O(n_{F_1}) \times n_x n_x n_\Delta &= O(n_{F_1} n_x n_x n_\Delta), \\ O(n_{F_1}) \times n_x n_q n_\Delta &= O(n_{F_1} n_x n_q n_\Delta) \end{aligned} \tag{D.179}$$

operations to calculate, as stipulated in equation (D.7). Apart from calculating partial derivative values, solving system (D.177) for all $i \in \{1, 2, \ldots, n_x\}$, $l \in \{1, 2, \ldots, n_q\}$, and $m \in \mathcal{I}_\Delta$ is equivalent in computational complexity to calculating

$$\begin{aligned} &\frac{\partial F_{i,m}}{\partial x_{j,m}}\frac{\partial x_{j,m}}{\partial q_l} \text{ for all } (i, j, l, m) \in \{1, \ldots, n_x\} \times \{1, \ldots, n_x\} \times \{1, \ldots, n_q\} \times \mathcal{I}_\Delta, \\ &\frac{\partial F_{i,m}}{\partial q_l} \text{ for all } (i, l, m) \in \{1, \ldots, n_x\} \times \{1, \ldots, n_q\} \times \mathcal{I}_\Delta, \end{aligned} \tag{D.180}$$

which, respectively, require

$$\begin{aligned} O(1) \circ O(I_0) \times n_x n_x n_q n_\Delta &= O(n_x n_x n_q n_\Delta), \\ O(I_0) \times n_x n_q n_\Delta &= O(n_x n_q n_\Delta) \end{aligned} \tag{D.181}$$

operations to calculate, where I_0 indicates values that have been calculated previously and $O(I_0) = 1$. Therefore, from complexity counts (D.179) and (D.181), in total, solving systems (D.177) for all $i \in \{1, 2, \ldots, n_x\}$, $l \in \{1, 2, \ldots, n_q\}$, and $m \in \mathcal{I}_\Delta$, the first order sensitivity equations for an initial value problem using the forward Euler method, requires

$$\begin{aligned} O(n_{F_1} n_x n_x n_\Delta) + O(n_{F_1} n_x n_q n_\Delta) + O(n_x n_x n_q n_\Delta) + O(n_x n_q n_\Delta) = \\ O\big(n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) \end{aligned} \tag{D.182}$$

operations. The forward Euler method applied to an initial value problem is the computationally least expensive explicit numerical solution method. Thus, in general, solving the first order sensitivity equations with an explicit numerical solution method requires

$$O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) \tag{D.183}$$

operations.

After calculating $\mathbf{x}$ and solving the first order sensitivity equations, calculating $\partial r(\mathbf{q})/\partial q_l$, of equation (D.175), for all $l \in \{1, 2, \ldots, n_q\}$ requires calculating partial derivative values,

$$\begin{aligned} \frac{\partial g_{j,k}}{\partial x_{m,k}} \text{ for all } (j,k,m) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_x\}, \\ \frac{\partial g_{j,k}}{\partial q_l} \text{ for all } (j,k,l) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\} \times \{1,\ldots,n_q\}, \\ \frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}} \text{ for all } (j,k) \in \{1,\ldots,n_y\} \times \{1,\ldots,n_t\}, \end{aligned} \tag{D.184}$$

which, respectively, require

$$\begin{aligned} O(n_{g_1}) \times n_y n_t n_x = O(n_{g_1} n_y n_t n_x), \\ O(n_{g_1}) \times n_y n_t n_q = O(n_{g_1} n_y n_t n_q), \\ O(1) \circ O(I_0) \times n_y n_t = O(n_y n_t) \end{aligned} \tag{D.185}$$

operations to calculate, as stipulated in Equations (D.3) and (D.4). After calculating $\mathbf{x}$, solving the first order sensitivity equations, and calculating partial derivative values, calculating $\partial r(\mathbf{q})/\partial q_l$, of equation (D.175), for all $l \in \{1, 2, \ldots, n_q\}$ is equivalent in computational complexity

to calculating

$$\begin{gathered}\frac{\partial g_{j,k}}{\partial x_{m,k}}\frac{\partial x_{m,k}}{\partial q_l} \text{ for all } (j,k,m,l) \in \\ \{1,\ldots,n_y\}\times\{1,\ldots,n_t\}\times\{1,\ldots,n_x\}\times\{1,\ldots,n_q\}, \\ \frac{\partial g_{j,k}}{\partial q_l} \text{ for all } (j,k,l) \in \{1,\ldots,n_y\}\times\{1,\ldots,n_t\}\times\{1,\ldots,n_q\}, \\ \frac{\partial d_{\mathrm{y}_{j,k}}}{\partial g_{j,k}}\cdot I_0 \text{ for all } (j,k) \in \{1,\ldots,n_y\}\times\{1,\ldots,n_t\},\end{gathered} \tag{D.186}$$

which, respectively, require

$$\begin{gathered}O(1)\circ O(I_0)\times n_y n_t n_x n_q = O(n_y n_t n_x n_q), \\ O(I_0)\times n_y n_t n_q = O(n_y n_t n_q), \\ O(1)\circ O(I_0)\times n_y n_t = O(n_y n_t)\end{gathered} \tag{D.187}$$

operations to calculate. Thus, from complexity counts (D.185) and (D.187), after calculating $\mathbf{x}$ and solving the first order sensitivity equations, calculating $\partial r(\mathbf{q})/\partial q_l$ for all $l \in \{1,2,\ldots,n_q\}$ requires

$$\begin{gathered}O(n_{g_1} n_y n_t n_x) + O(n_{g_1} n_y n_t n_q) + O(n_y n_t) + O(n_y n_t n_x n_q) + O(n_y n_t n_q) + \\ O(n_y n_t) = O\big(n_y n_t (n_{g_1} n_x + n_{g_1} n_q + n_x n_q)\big)\end{gathered} \tag{D.188}$$

operations. Therefore, from complexity counts (D.174), (D.183), and (D.188), in total, calculating $\partial r(\mathbf{q})/\partial q_l$ for all $l \in \{1,2,\ldots,n_q\}$ requires

$$\begin{gathered}O(n_f n_x n_\Delta) + O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + \\ O\big(n_y n_t (n_{g_1} n_x + n_{g_1} n_q + n_x n_q)\big) = \\ O\big(n_f n_x n_\Delta + n_y n_t n_{g_1}(n_x + n_q)\big) + O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big)\end{gathered} \tag{D.189}$$

operations with an explicit numerical solution method, as $n_y \leq n_x$ and $n_t \leq n_\Delta$.

For implicit numerical solution methods, $f_{i,k}(\mathbf{t},\mathbf{q},\mathbf{x}) = 0$ is an implicit system of equations in $\mathbf{x}$ and discretized sensitivity equations are implicit systems of equations in discretized sensitivity values, $\partial x_{i,m}/\partial q_l$. Generally, $f_{i,k}(\mathbf{t},\mathbf{q},\mathbf{x}) = 0$ is a nonlinear system of equations that is solved numerically using Newton's method. Each iteration of Newton's method to solve $f_{i,k}(\mathbf{t},\mathbf{q},\mathbf{x}) = 0$ requires calculating the values of $f_{i,k}(\mathbf{t},\mathbf{q},\mathbf{x})$ and the values of first order partial derivatives of $f_{i,k}(\mathbf{t},\mathbf{q},\mathbf{x})$ with respect to $x_{l,m}$, for all $i \in \{1,2,\ldots,n_x\}$, $k \in \mathcal{I}_\Delta$, $l \in \{1,2,\ldots,n_x\}$, and $m \in \mathcal{I}_\Delta$. Calculating

$$f_{i,k}(\mathbf{t},\mathbf{q},\mathbf{x}) \text{ for all } (i,k) \in \{1,2,\ldots,n_x\}\times\mathcal{I}_\Delta \tag{D.190}$$

requires

$$O(n_f) \times n_x n_\Delta = O(n_f n_x n_\Delta) \tag{D.191}$$

operations, as stipulated in equation (D.5). $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x})$ depend on $x_{l,m}$ for only a small fraction of k in $\mathcal{I}_\Delta$, at k in $\mathcal{I}_{\Delta_m} \subset \mathcal{I}_\Delta$. Calculating

$$\frac{\partial f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x})}{\partial x_{l,m}} \text{ for all } (i,k,l,m) \in \{1,\ldots,n_x\} \times \mathcal{I}_{\Delta_m} \times \{1,\ldots,n_x\} \times \mathcal{I}_\Delta \tag{D.192}$$

requires

$$O(n_{f_1}) \times n_x n_\delta n_x n_\Delta = O(n_{f_1} n_x n_\delta n_x n_\Delta) \tag{D.193}$$

operations, as stipulated in equation (D.5). Additionally, each iteration of Newton's method to solve $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ requires solving matrix equations, requiring a total of $O(n_M)$ operations. Thus, in conjunction with complexity counts (D.191) and (D.193), each iteration of Newton's method to solve $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ requires

$$\begin{gathered} O(n_x n_\Delta n_f) + O(n_{f_1} n_x n_\delta n_x n_\Delta) + O(n_M) = \\ O\big(n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_M\big) \end{gathered} \tag{D.194}$$

operations. Solving $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ requires $O(n_N)$ iterations of Newton's method. Thus, for an implicit numerical solution method, numerically solving $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ to determine $\mathbf{x}$ requires

$$O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M\big) \tag{D.195}$$

operations.

The sensitivity equations are generated by applying the chain rule to the differential equation system, and are thus linear in sensitivity values, $\partial x_i / \partial q_l$. As such, discretized sensitivity equations are generally linear in discretized sensitivity values, $\partial x_{i,m} / \partial q_l$. Thus, beyond the calculations required to solve the discretized sensitivity equations with an explicit numerical solution method, solving the discretized sensitivity equations with an implicit numerical solution method requires solving matrix equations. First order discretized sensitivity equations with respect to q_l are identical in size to $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$. Thus, calculating matrix equations in solving first order discretized sensitivity equations with respect to q_l requires a total of $O(n_M)$ operations. As such, calculating matrix equations in solving first order discretized sensitivity equations with respect to q_l for all $l \in \{1, 2, \ldots, n_q\}$ requires a total of $O(n_q n_M)$ operations. Therefore, in general, in conjunction with complexity count (D.183), solving the first order sensitivity equations with an implicit numerical solution method requires

$$O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + O(n_q n_M) \tag{D.196}$$

operations. From complexity count (D.188), after calculating $\mathbf{x}$ and solving the first order sensitivity equations, calculating $\partial r(\mathbf{q})/\partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$ requires $O(n_y n_t (n_{g_1} n_x + n_{g_1} n_q + n_x n_q))$ operations. Therefore, from complexity counts (D.195), (D.196), and (D.188), in total, calculating $\partial r(\mathbf{q})/\partial q_l$ for all $l \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{gathered} O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M\big) + O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) + \\ O(n_q n_M) + O\big(n_y n_t (n_{g_1} n_x + n_{g_1} n_q + n_x n_q)\big) = \\ O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_M (n_q + n_N) + n_y n_t n_{g_1} (n_x + n_q)\big) + \\ O\big(\geq n_x n_\Delta (n_{F_1} n_x + n_{F_1} n_q + n_x n_q)\big) \end{gathered} \tag{D.197}$$

operations with an implicit numerical solution method, as $n_y \leq n_x$ and $n_t \leq n_\Delta$.

Alternatively, I can approximate partial derivatives of $r(\mathbf{q})$ with respect to parameters by finite differences, rather than by solving the sensitivity equations. Most simply,

$$\frac{\partial r(\mathbf{q})}{\partial q_l} \approx \frac{r(\mathbf{q} + h_l \mathbf{e}_l) - r(\mathbf{q})}{h_l}, \tag{D.198}$$

where $\mathbf{e}_l$ is the l^{th} standard basis vector and h_l is some small perturbation in parameter q_l.

After calculating $\mathbf{x}$, calculating $r(\mathbf{q})$ requires calculating

$$\frac{1}{n_y} \sum_{j=1}^{n_y} \sum_{k=1}^{n_t} d_{\mathrm{y}_{j,k}}\big(g_{j,k}(\mathbf{q}, \mathbf{x})\big), \tag{D.199}$$

which is equivalent in computational complexity to calculating

$$d_{\mathrm{y}_{j,k}}\big(g_{j,k}(\mathbf{q}, \mathbf{x})\big) \text{ for all } (j,k) \in \{1, \ldots, n_y\} \times \{1, \ldots, n_t\}, \tag{D.200}$$

which requires

$$O(1) \circ O(n_g) \times n_y n_t = O(n_g n_y n_t) \tag{D.201}$$

operations to calculate, as stipulated in Equations (D.3) and (D.4). Thus, from complexity counts (D.174) and (D.201), calculating $r(\mathbf{q})$ requires

$$O(n_f n_x n_\Delta) + O(n_g n_y n_t) = O(n_f n_x n_\Delta + n_g n_y n_t) \tag{D.202}$$

operations with an explicit numerical solution method, and, from complexity counts (D.195) and (D.201), calculating $r(\mathbf{q})$ requires

$$\begin{gathered} O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M\big) + O(n_g n_y n_t) = \\ O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big) \end{gathered} \tag{D.203}$$

operations with an implicit numerical solution method.

Approximating $\partial r(\mathbf{q})/\partial q_l$, as in equation (D.198), for all $l \in \{1, 2, \ldots, n_q\}$ is equivalent in computational complexity to calculating

$$\begin{gathered} r(\mathbf{q} + h_l \mathbf{e}_l) \text{ for all } l \in \{1, 2, \ldots, n_q\}, \\ r(\mathbf{q}), \end{gathered} \tag{D.204}$$

which, from complexity count (D.202), for an explicit numerical solution method, respectively, require

$$\begin{gathered} O(n_f n_x n_\Delta + n_g n_y n_t) \times n_q = O\big(n_q(n_f n_x n_\Delta + n_g n_y n_t)\big), \\ O(n_f n_x n_\Delta + n_g n_y n_t) \end{gathered} \tag{D.205}$$

operations to calculate, and, from complexity count (D.203), for an implicit numerical solution method, respectively, require

$$\begin{gathered} O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big) \times n_q = \\ O\big(n_N n_x n_\Delta n_q (n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q + n_g n_y n_t n_q\big), \\ O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big) \end{gathered} \tag{D.206}$$

operations to calculate. Thus, for an explicit numerical solution method, approximating $\partial r(\mathbf{q})/\partial q_l$ by finite difference for all $l \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{gathered} O\big(n_q(n_f n_x n_\Delta + n_g n_y n_t)\big) + O(n_f n_x n_\Delta + n_g n_y n_t) = \\ O\big(n_q(n_f n_x n_\Delta + n_g n_y n_t)\big) \end{gathered} \tag{D.207}$$

operations, and for an implicit numerical solution method, approximating $\partial r(\mathbf{q})/\partial q_l$ by finite difference for all $l \in \{1, 2, \ldots, n_q\}$ and $m \in \{1, 2, \ldots, n_q\}$ requires

$$\begin{gathered} O\big(n_N n_x n_\Delta n_q (n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q + n_g n_y n_t n_q\big) + \\ O\big(n_N n_x n_\Delta (n_f + n_{f_1} n_x n_\delta) + n_N n_M + n_g n_y n_t\big) = \\ O\big(n_N n_x n_\Delta n_q (n_f + n_{f_1} n_x n_\delta) + n_N n_M n_q + n_g n_y n_t n_q\big) \end{gathered} \tag{D.208}$$

operations. □

D.3 Comparison of Computational Complexities

D.3.1 Complexity Assumptions for Comparison

In Theorems 7, 8, 9, and 10, I calculate computational complexities with general n_g, n_{g_1}, n_{g_2}, n_p, n_F, n_{F_1}, n_{F_2}, n_f, n_{f_1}, n_{f_2}, n_δ, n_σ, n_N, and n_M. To compare computational complexities, I

consider more specific n_g, n_{g_1}, n_{g_2}, n_p, n_F, n_{F_1}, n_{F_2}, n_f, n_{f_1}, n_{f_2}, n_δ, n_σ, n_N, and n_M. Often, observable-state functions, $g_{j,k}(\mathbf{p}, \mathbf{x})$, are linear combinations of state values. Thus, I consider

$$\begin{aligned} O(n_g) &= O(n_x), \\ O(n_{g_1}) &= O(1), \\ O(n_{g_2}) &= O(0). \end{aligned} \tag{D.209}$$

Often, a discretized differential equation, $F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$, is a sum of nonlinear parameter and discretized state-value combinations, with $O(n_x)$ nonlinear combinations of $O(n_x)$ parameters in interconnected systems, for each $i \in \{1, 2, \ldots, n_x\}$. Thus, I consider

$$\begin{aligned} O(n_p) &= O(n_x^2), \\ O(n_F) &= O(n_x), \\ O(n_{F_1}) &= O(1), \\ O(n_{F_2}) &= O(1). \end{aligned} \tag{D.210}$$

Generally, calculating numerical discretizations, $f_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$, requires a similar number of operations as calculating discretized differential equation values, $F_{i,k}(\mathbf{t}, \mathbf{p}, \mathbf{x})$. Thus, I consider

$$\begin{aligned} O(n_f) &= O(n_x), \\ O(n_{f_1}) &= O(1), \\ O(n_{f_2}) &= O(1). \end{aligned} \tag{D.211}$$

Generally, the number of elements in $\mathcal{I}_{\Delta_m}$ is similar to the order of the differential equation, ordinarily of $O(1)$. Therefore, I consider

$$O(n_\delta) = O(1). \tag{D.212}$$

operations. I consider $O(1)$ line-search test points in each iteration of descent and $O(1)$ iterations of Newton's method to solve $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ for all $i \in \{1, 2, \ldots, n_x\}$ and $k \in \mathcal{I}_\Delta$. Thus,

$$\begin{aligned} O(n_\sigma) &= O(1), \\ O(n_N) &= O(1), \end{aligned} \tag{D.213}$$

Often, initial conditions and boundary values are not fixed, and are fitted along with model parameters to data. Thus, for n_i initial points in $\mathcal{I}_\Delta$ and n_b boundary points in $\mathcal{I}_\Delta$, I consider

$$O(n_q) = O(n_p + n_x n_i + n_x n_b). \tag{D.214}$$

For clarity, I note that $n_i = 1$ and $n_b = 0$ for initial value ordinary differential equations and $n_i = 0$ and $n_b = 2$ for boundary value ordinary differential equations.

Calculating $\mathbf{x}_{n+1}$, the $n_x n_\Delta$ state values, $x_{l,m}$ for all $l \in \{1, 2, \ldots, n_x\}$ and $m \in \mathcal{I}_\Delta$, in the $n + 1^{\text{th}}$ iteration of Newton's method applied to solving the system $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x}) = 0$ for all $i \in \{1, 2, \ldots, n_x\}$ and $k \in \mathcal{I}_\Delta$, requires calculating

$$J(\mathbf{x}_n)(\mathbf{x}_{n+1} - \mathbf{x}_n) = -\mathbf{f}(\mathbf{x}_n), \tag{D.215}$$

where $J(\mathbf{x})$ is the $n_x n_\Delta \times n_x n_\Delta$ Jacobian matrix of $\mathbf{f}(\mathbf{x})$. For locally implicit numerical solution methods, such as the backward Euler method for ordinary differential equations, matrix equation (D.215) is separable into n_Δ submatrix equations, each of size $n_x \times n_x$. For interconnected models, generally, $\partial f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x})/\partial x_{l,m} \neq 0$ if $m = k$. Thus, for interconnected models, each $n_x \times n_x$ submatrix of $J(\mathbf{x}_n)$ is a full matrix, and solving each of the n_Δ full submatrix equations requires $O(n_x^3)$ operations using Gaussian elimination. For globally implicit numerical solution methods, matrix equation (D.215) is not separable into smaller matrix equations. Although, $J(\mathbf{x}_n)$ is large and sparse, so solving matrix equation (D.215) with an iterative method, such as the generalized minimal residual method, is computationally more efficient than solving matrix equation (D.215) with a direct method, such as Gaussian elimination. $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x})$ depends on $x_{l,m}$ for only a small fraction of k in $\mathcal{I}_\Delta$, at k in $\mathcal{I}_{\Delta_m} \subset \mathcal{I}_\Delta$. Thus, for interconnected models, $f_{i,k}(\mathbf{t}, \mathbf{q}, \mathbf{x})$ depends on $O(n_\delta n_x)$ values of $x_{l,m}$, and $J(\mathbf{x}_n)$ contains $O(n_\Delta n_x \times n_\delta n_x) = O(n_\Delta n_x^2 n_\delta)$ nonzero elements. In each iteration of the generalized minimal residual method, multiplication by $J(\mathbf{x}_n)$ requires $O(n_\Delta n_x^2 n_\delta)$ operations. The generalized minimal residual method may may require up to $O(n_\Delta n_x)$ iterations to exactly solve matrix equation (D.215), but will generally converge in significantly fewer iterations. Therefore, for all implicit numerical solution methods, I consider

$$O(n_M) = O(\geq n_\Delta n_x^3). \tag{D.216}$$

D.3.2 Comparison of Computational Complexities with Assumptions

Computational Complexity of $r(\mathbf{p}, \mathbf{x}; \lambda)$ Descent with Assumptions

From assumptions (D.209), (D.210), (D.211), (D.212), and (D.213) and Theorem 7, an iteration of $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent requires

$$\begin{gathered} O\big(n_\Delta(n_x n_y + n_x n_x)\big) + O\big(n_\Delta n_p(n_y + n_x)\big) + O\big(n_\Delta n_x(n_y + n_x)\big) = \\ n_\Delta n_p n_x = n_\Delta n_x^3 \end{gathered} \tag{D.217}$$

operations, as $n_y \leq n_x \leq n_p$.

Computational Complexity of $r(\mathbf{q})$ Descent with Assumptions

From assumptions (D.209), (D.210), (D.211), (D.212), (D.213), (D.214), and (D.216) and Theorem 8, an iteration of $r(\mathbf{q})$ descent requires

$$\begin{gathered} O\big(n_x n_x n_\Delta + n_x n_y n_t + n_y n_t (n_x + n_q)\big) + \\ O\big(\geq n_x n_\Delta (n_x + n_q + n_x n_x + n_x n_q + n_x n_x n_q)\big) = \\ O(\geq n_\Delta n_q n_x^3) = O\big(\geq n_\Delta (n_x + n_i + n_b) n_x^4\big) \end{gathered} \tag{D.218a}$$

operations with an explicit numerical solution method, as $n_y \leq n_x$ and $n_t \leq n_\Delta$; an iteration of $r(\mathbf{q})$ descent requires

$$\begin{gathered} O\big(n_x n_\Delta n_x + n_x n_y n_t\big) + O\big(n_M n_q + n_y n_t (n_x + n_q)\big) + \\ O\big(\geq n_x n_\Delta (n_x + n_q + n_x n_x + n_x n_q + n_x n_x n_q)\big) = \\ O(\geq n_\Delta n_q n_x^3) + O(n_M n_q) = \\ O\big(\geq n_\Delta (n_x + n_i + n_b) n_x^4\big) + O\big(\geq n_\Delta n_x^4 (n_x + n_i + n_b)\big) = \\ O\big(\geq n_\Delta (n_x + n_i + n_b) n_x^4\big) \end{gathered} \tag{D.218b}$$

operations with an implicit numerical solution method, as $n_y \leq n_x$ and $n_t \leq n_\Delta$; an iteration of $r(\mathbf{q})$ descent requires

$$O\big(n_q (n_x n_x n_\Delta + n_x n_y n_t)\big) = O(n_\Delta n_q n_x^2) = O\big(n_\Delta (n_x + n_i + n_b) n_x^3\big) \tag{D.218c}$$

operations with an explicit numerical solution method and partial derivative approximation by finite difference, as $n_y \leq n_x$ and $n_t \leq n_\Delta$; and an iteration of $r(\mathbf{q})$ descent requires

$$\begin{gathered} O\big(n_q (n_x n_\Delta n_x + n_M + n_x n_y n_t)\big) = O(n_\Delta n_q n_x^2) + O(n_M n_q) = \\ O\big(n_\Delta (n_x + n_i + n_b) n_x^3\big) + O\big(\geq n_\Delta n_x^4 (n_x + n_i + n_b)\big) = \\ O\big(\geq n_\Delta (n_x + n_i + n_b) n_x^4\big) \end{gathered} \tag{D.218d}$$

operations with an implicit numerical solution method and partial derivative approximation by finite difference, as $n_y \leq n_x$ and $n_t \leq n_\Delta$. Comparing computational counts (D.217) and (D.218), for an explicit numerical solution method, an iteration of $r(\mathbf{q})$ descent requires at least $n_x + n_i + n_b$ times as many operations as an iteration $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent, and, for an implicit numerical solution method, an iteration of $r(\mathbf{q})$ descent requires at least $(n_x + n_i + n_b) n_x$ times as many operations as an iteration $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent. For ODEs, where $n_i + n_b \leq 2$, $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent is computationally more efficient than $r(\mathbf{q})$ descent, and the difference in computational efficiency increases with an increasing number of model states, markedly for implicit numerical solution methods. Generally, in PDE models of data, the number of initial points and/or the number of boundary points greatly exceeds the number of states, $n_x << n_i + n_b$. Thus, for PDEs,

$r(\mathbf{p}, \mathbf{x}; \lambda)$ descent is computationally far more efficient than $r(\mathbf{q})$ descent, and the difference in computational efficiency grows with an increasing number of data points and/or model states, markedly for implicit numerical solution methods.

Computational Complexity of Newton's Method to Minimize $r(\mathbf{q})$ with Assumptions

From assumptions (D.209), (D.210), (D.211), (D.212), (D.213), (D.214), and (D.216) and Theorem 9, an iteration of $r(\mathbf{q})$ minimization by Newton's method requires

$$\begin{gathered} O\big(n_x n_x n_\Delta + n_y n_t (n_x + n_q)\big) + \\ O\big(\geq n_x n_\Delta (n_x + n_q + n_x n_x + n_x n_q + n_q n_q + n_x n_x n_q + n_x n_q n_q)\big) = \\ O(\geq n_\Delta n_q^2 n_x^2) = O\big(\geq n_\Delta (n_x + n_i + n_b)^2 n_x^4\big) \end{gathered} \tag{D.219a}$$

operations with an explicit numerical solution method, as $n_y \leq n_x \leq n_q$ and $n_t \leq n_\Delta$; an iteration of $r(\mathbf{q})$ minimization by Newton's method requires

$$\begin{gathered} O\big(n_x n_\Delta n_x + n_M n_q n_q + n_y n_t (n_x + n_q)\big) + \\ O\big(\geq n_x n_\Delta (n_x + n_q + n_x n_x + n_x n_q + n_q n_q + n_x n_x n_q + n_x n_q n_q)\big) = \\ O(\geq n_\Delta n_q^2 n_x^2) + O(n_M n_q^2) = \\ O\big(\geq n_\Delta (n_x + n_i + n_b)^2 n_x^4\big) + O\big(\geq n_\Delta n_x^5 (n_x + n_i + n_b)^2\big) = \\ O\big(\geq n_\Delta (n_x + n_i + n_b)^2 n_x^5\big) \end{gathered} \tag{D.219b}$$

operations with an implicit numerical solution method, as $n_y \leq n_x \leq n_q$ and $n_t \leq n_\Delta$; an iteration of $r(\mathbf{q})$ minimization by Newton's method requires

$$\begin{gathered} O\big(n_q n_q (n_x n_x n_\Delta + n_x n_y n_t)\big) = O(n_\Delta n_q^2 n_x^2) = \\ O\big(n_\Delta (n_x + n_i + n_b)^2 n_x^4\big) \end{gathered} \tag{D.219c}$$

operations with an explicit numerical solution method and partial derivative approximation by finite difference, $n_y \leq n_x$ and $n_t \leq n_\Delta$; and an iteration of $r(\mathbf{q})$ minimization by Newton's method requires

$$\begin{gathered} O(n_x n_\Delta n_q n_q n_x + n_M n_q n_q + n_x n_y n_t n_q n_q) = O(n_\Delta n_q^2 n_x^2) + O(n_M n_q^2) = \\ O\big(n_\Delta (n_x + n_i + n_b)^2 n_x^4\big) + O\big(\geq n_\Delta n_x^5 (n_x + n_i + n_b)^2\big) = \\ O\big(\geq n_\Delta (n_x + n_i + n_b)^2 n_x^5\big) \end{gathered} \tag{D.219d}$$

operations with an implicit numerical solution method and partial derivative approximation by finite difference, as $n_y \leq n_x$ and $n_t \leq n_\Delta$. Comparing computational counts (D.217) and (D.219), for an explicit numerical solution method, an iteration of $r(\mathbf{q})$ minimization by Newton's method

requires at least $(n_x + n_i + n_b)^2 n_x$ times as many operations as an iteration $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent, and, for an implicit numerical solution method, an iteration of $r(\mathbf{q})$ minimization by Newton's method requires at least $(n_x + n_i + n_b)^2 n_x^2$ times as many operations as an iteration $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent. Newton's method generally converges quadratically. For ODEs, where $n_i + n_b \leq 2$, with few model states, superior convergence of Newton's method may compensate for its relatively large computational burden, but, with an increasing number of model states, $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent becomes increasingly more computationally efficient than $r(\mathbf{q})$ minimization by Newton's method, markedly for implicit numerical solution methods. For PDEs, where $n_x << n_i + n_b$, $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent is computationally far more efficient than $r(\mathbf{q})$ minimization by Newton's method, and the difference in computational efficiency grows with an increasing number of data points and/or model states, markedly for implicit numerical solution methods.

Computational Complexity of Gradient-Based Methods to Minimize $r(\mathbf{q})$ with Assumptions

To minimize $r(\mathbf{q})$, gradient-based methods, such as gradient descent, the Gauss-Newton method, Levenberg-Marquardt method, and quasi-Newton methods, require calculating all first order partial derivatives of $r(\mathbf{q})$. From assumptions (D.209), (D.210), (D.211), (D.212), (D.213), (D.214), and (D.216) and Theorem 10, calculating all first order partial derivatives of $r(\mathbf{q})$ requires

$$\begin{gathered} O\big(n_x n_x n_\Delta + n_y n_t (n_x + n_q)\big) + O\big(\geq n_x n_\Delta (n_x + n_q + n_x n_q)\big) = \\ O(\geq n_\Delta n_q n_x^2) = O\big(\geq n_\Delta (n_x + n_i + n_b) n_x^3\big) \end{gathered} \tag{D.220a}$$

operations with an explicit numerical solution method, as $n_y \leq n_x$ and $n_t \leq n_\Delta$; calculating all first order partial derivatives of $r(\mathbf{q})$ requires

$$\begin{gathered} O\big(n_x n_\Delta n_x + n_M n_q + n_y n_t (n_x + n_q)\big) + O\big(\geq n_x n_\Delta (n_x + n_q + n_x n_q)\big) = \\ O(\geq n_\Delta n_q n_x^2) + O(n_M n_q) = \\ O\big(\geq n_\Delta (n_x + n_i + n_b) n_x^3\big) + O\big(\geq n_\Delta n_x^4 (n_x + n_i + n_b)\big) = \\ O\big(\geq n_\Delta (n_x + n_i + n_b) n_x^4\big) \end{gathered} \tag{D.220b}$$

operations with an implicit numerical solution method, as $n_y \leq n_x$ and $n_t \leq n_\Delta$; approximating all first order partial derivatives of $r(\mathbf{q})$ by finite difference requires

$$\begin{gathered} O\big(n_q (n_x n_x n_\Delta + n_x n_y n_t)\big) = O(n_\Delta n_q n_x^2) = \\ O\big(n_\Delta (n_x + n_i + n_b) n_x^3\big) \end{gathered} \tag{D.220c}$$

operations with an explicit numerical solution method, as $n_y \leq n_x$ and $n_t \leq n_\Delta$; and approximating all first order partial derivatives of $r(\mathbf{q})$ by finite difference requires

$$\begin{gathered} O(n_x n_\Delta n_q n_x + n_M n_q + n_x n_y n_t n_q) = O(n_\Delta n_q n_x^2) + O(n_M n_q) = \\ O\big(n_\Delta (n_x + n_i + n_b) n_x^3\big) + O\big(\geq n_\Delta n_x^4 (n_x + n_i + n_b)\big) = \\ O\big(\geq n_\Delta (n_x + n_i + n_b) n_x^4\big) \end{gathered} \tag{D.220d}$$

operations with an implicit numerical solution method, as $n_y \leq n_x$ and $n_t \leq n_\Delta$. Comparing computational counts (D.217) and (D.220), for an explicit numerical solution method, calculating all first order partial derivatives of $r(\mathbf{q})$ requires at least $n_x + n_i + n_b$ times as many operations as an iteration $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent, and, for an implicit numerical solution method, calculating all first order partial derivatives of $r(\mathbf{q})$ requires at least $(n_x + n_i + n_b)n_x$ times as many operations as an iteration $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent. In some cases, gradient-based methods that approximate the Hessian using first order partial derivative values, such as the Gauss-Newton method, Levenberg-Marquardt method, and quasi-Newton methods, may converge somewhat faster than descent. In such cases, for ODEs, where $n_i + n_b \leq 2$, with few model states, superior convergence of gradient-based methods may compensate for relatively larger computational burdens, but, with an increasing number of model states, $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent becomes increasingly more computationally efficient than $r(\mathbf{q})$ minimization by gradient-based methods, markedly for implicit numerical solution methods. For PDEs, where $n_x << n_i + n_b$, $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent is computationally far more efficient than $r(\mathbf{q})$ minimization by gradient-based methods, and the difference in computational efficiency grows with an increasing number of data points and/or model states, markedly for implicit numerical solution methods.

Conclusion

Overall, from assumptions (D.209), (D.210), (D.211), (D.212), (D.213), (D.214), and (D.216) and Theorems 7, 8, 9, and 10, $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent is computational more efficient than $r(\mathbf{q})$ minimization. More specifically, for ODEs, with few model states, $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent may be somewhat more computationally efficient than $r(\mathbf{q})$ minimization, and, with an increasing number of model states, $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent becomes increasingly more computationally efficient than $r(\mathbf{q})$ minimization, markedly for implicit numerical solution methods. Also, for PDEs, $r(\mathbf{p}, \mathbf{x}; \lambda)$ descent is computationally far more efficient than $r(\mathbf{q})$ minimization, and the difference in computational efficiency grows with an increasing number of data points and/or model states, markedly for implicit numerical solution methods.

Appendix E

Details of Testing the Homotopy-Minimization Method for Parameter Estimation in Differential Equations

E.1 Implementation of Overlapping-Niche Descent for Forms of the Bonny Model

Here, I describe details pertaining to the implementation of overlapping-niche descent for forms of the Bonny model. I describe related structural components of overlapping-niche descent in Section 3.4.

E.1.1 Generating Random Parameters and State Values

Initially in overlapping-niche descent, I randomly generate parameters and state values. Also, as discussed in Section C.1, throughout overlapping-niche descent, I randomly generate parameters and state values in random offspring. Given no prior parameter value estimates, I randomly generate rate parameters over a broad range of scales:

$$p \sim u_p \cdot 10^{U(-6,1)} \text{ for all } p \in \{\omega_D, \omega_{dD}, \omega_E, \omega_{ed}, \omega_{de,m}, \omega_{de,c}, \omega_e\}, \tag{E.1}$$

where $U(a, b)$ is the uniform probability distribution over the interval (a, b), and u_p is the units of parameter p. I expect $c_{\max}$ to be within one or two orders of magnitude of the maximal MinD data value, $D_{\max}$. Thus I randomly generate $c_{\max}$ such that

$$c_{\max} \sim D_{\max} \cdot 10^{U(0,2)}. \tag{E.2}$$

Naively, I guess that fitted diffusion coefficients are within one or two orders of magnitude of 10 $\mu m^2\ s^{-1}$. Thus, I randomly generate diffusion coefficients such that

$$p \sim 10\ \mu m^2\ s^{-1} \cdot 10^{U(-2,2)} \text{ for all } p \in \{D_d, D_{de}, D_e\}. \tag{E.3}$$

For reference, the parameter values used to generate synthetic data are shown in Table 3.1. I randomly generate state values so that observable state values match observed or interpolated data exactly. Thus, for MinD and MinE observed or interpolated data values, D and E, with c_e as a free state, I randomly generate state values such that

$$\begin{aligned} c_e &\sim U(\max\{0, E - D\}, E), \\ c_{de} &= E - c_e, \\ c_d &= D - E + c_e. \end{aligned} \tag{E.4}$$

E.1.2 Parents and Offspring

The function of parents and offspring in overlapping-niche descent is described in Section C.1. Accordingly, to the i^{th} niche in generation g, I allocate one sustained parent, $\hat{n}_i = 1$, one high momentum offspring, $\check{n}^m_{g,i} = 1$, one cross-niche offspring, $\check{n}^c_{g,i} = 1$, and one random offspring, $\check{n}^r_{g,i} = 1$, for each $i \in \{1, 2, \ldots, 101\}$ and each generation of overlapping-niche descent, $g \geq 1$. In the first two generations of overlapping-niche descent, $g \leq 2$, I allocate two sexual offspring to each niche, $\check{n}^s_{g,i} = 2$ for all $i \in \{1, 2, \ldots, 101\}$. After the second generation of overlapping-niche descent, I adaptively change the number of sexual offspring that I allocate to each niche, enlarging less convergent niches and shrinking more convergent niches for greater efficiency in optimization. Specifically, I allocate one sexual offspring to the i^{th} niche, and randomly allocate the remaining 101 sexual offspring to the i^{th} niche with probability proportional to $\Delta r_{g,i,1}$, the measure of convergence in the (first) parent space of the i^{th} niche in generation g, as defined in equation (C.1), for each $i \in \{1, 2, \ldots, 101\}$ and $g > 2$.

E.1.3 Selection and Random Perturbation

I choose the natural default value for the selection strength parameter, $q_{\text{fit}} = 1$, for q_{fit} as described in Section C.1. For a sexual offspring that inherits parameter p from individual $(\mathbf{p}_{g,i,j}, \mathbf{x}_{g,i,j})$, I perturb the value of the parental parameter, $\hat{p}$, such that

$$p \sim \left(\hat{p} \cdot 10^{N\left(0, \max\{\Delta r_{g,i,j}, 10^{-2}\}^2\right)} \mid p \geq p_{\min}\right), \tag{E.5}$$

where $N(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2, $\Delta r_{g,i,j}$ is the measure of convergence in the j^{th} parent space of the i^{th} niche in generation g, as defined in equation (C.1), and $p_{\min}$ is the restricted lower bound on parameter p as discussed in Section 3.4.2. A standard deviation of $\max\{\Delta r_{g,i,j}, 10^{-2}\}$ ensures some small but significant perturbation in parameter p when $\Delta r_{g,i,j}$ is small. Similarly, for a sexual offspring that inherits state value x from individual $(\mathbf{p}_{g,i,j}, \mathbf{x}_{g,i,j})$, I perturb the value of the parental state value, $\hat{x}$, such that

$$x \sim \left(\hat{x} + \hat{x} \cdot N\left(0, \max\{\Delta r_{g,i,j}, 10^{-2}\}^2\right) \mid x \geq 0\right). \tag{E.6}$$

Details pertaining to sexual offspring are described in Section C.1.

E.1.4 Dykstra's Method

As discussed in Section C.2.3, during overlapping-niche descent, I project parameters and state values onto a restricted domain using Dykstra's method. For forms of the Bonny model, restrictions on parameters and state values, linear inequalities (3.7), (3.8), and (3.9), are all mutually independent. Thus, Dykstra's method, as discussed in Section C.2.3, will converge in one projection cycle, when projecting parameters and state values onto the domain bounded by (3.7), (3.8), and (3.9). As such, for Dykstra's method, I have no need to define the relative termination tolerance, ε_c or the absolute termination tolerance, $\varepsilon_{\bar{c}}$.

E.1.5 Initial values, Termination, Prolongation, and Computation

I choose the initial gradient scaling value $s_{i,0} = 0$, for all i in the indexed set of all parameters and state values. Details pertaining to $s_{i,0}$ are described in Section C.2.1. I choose the maximum number of strict descent iterations to be relatively but not excessively large, $n_{\max} = 10^4$, to ensure sufficient convergence to a local minimum of $r(\mathbf{p}, \mathbf{x}; \lambda)$ while avoiding overburdensome computation. I choose a very small contraction termination tolerance, $\varepsilon_\sigma = 10^{-30}$, and a very small relative-change termination tolerance, $\varepsilon_r = 10^{-30}$, to continue accelerated descent through $n_{\max}$ strict descent iteration unless local minimization is essentially complete. Details pertaining to $n_{\max}$, ε_σ, and ε_r are described in Section C.2.2. For descent prolongation, I choose: $\breve{\sigma} = 1$, for non-stringent descent prolongation, $m_{\text{pro}} = 10^3$, a factor of 10 less than $n_{\max}$; $\hat{n}_{\text{pro}} = n_{\max} = 10^4$; and $\check{n}_{\text{pro}} = n_{\max} = 10^4$. Details pertaining to $\breve{\sigma}$, m_{pro}, $\hat{n}_{\text{pro}}$, and $\check{n}_{\text{pro}}$ are described in Section C.3. I choose the overlapping-niche descent termination tolerance to be relatively but not exceedingly small, $\varepsilon_{\Delta r} = 10^{-3}$, for reliable convergence in all niches while avoiding an excessive number of overlapping-niche descent generations. Details pertaining to $\varepsilon_{\Delta r}$ are described in Section C.1. I compute genetic algorithm calculations using *MATLAB*. I compute accelerated descent calculations in parallel using *C++* on the Calcul Québec server Guillimin, which uses Intel Xeon X5650 Westmere processors and has 6 cores per CPU.

E.2 Details of SNSD

Here, I describe details of SNSD, single-niche solution descent. SNSD optimizes over parameters and initial conditions to minimize $r_{\text{y}}(\mathbf{p}, \mathbf{x})$ with numerical solution values $\mathbf{x}$. SNSD follows the genetic algorithm of overlapping-niche descent as described in Section C.1, but consists of a single niche. To be consistent with overlapping-niche descent as described in Section 3.4.3, SNSD sustains 101 parents, 101 one high momentum offspring, 303 sexual offspring, and 101 random offspring. SNSD follows the accelerated descent routine of overlapping-niche descent as described in Section C.2, except state values are determined by numerically solving the differential equation system. Also, state values are implicit functions of parameters and initial

conditions. So, partial derivatives of $r_{\mathrm{y}}(\mathbf{p}, \mathbf{x})$ with respect to parameters and initial conditions are calculated in part by numerically solving the sensitivity Equations (D.52) to determine partial derivatives of state values with respect to parameters and initial conditions.

E.3 Details Pertaining to the Implementation of Overlapping-Niche Descent in Practice with the Full Bonny Model

Here, I continue the discussion of Section 3.8, to explicate details pertaining to the implementation of overlapping-niche descent in practice. My discussion follows overlapping-niche descent in the fitting of the full Bonny model, as defined in Equation 3.1, to the synthetic traveling-wave-emergence data, as shown in Figure 3.3, on a uniform grid with a grid refinement factor of 1, $n_{\Delta_t} n_{\Delta_s} {n_t}^{-1} {n_s}^{-1} = 1$ for n_{Δ_t} and n_{Δ_s} the number or temporal and spatial grid points and n_t and n_s the number of temporal and spatial data points.

E.3.1 Selection in Overlapping-Niche Descent

As discussed in Section C.1, overlapping-niche descent employs various types of offspring. To illustrate how offspring types contribute to convergence in overlapping-niche descent, I map selection from offspring types in Figure E.1.

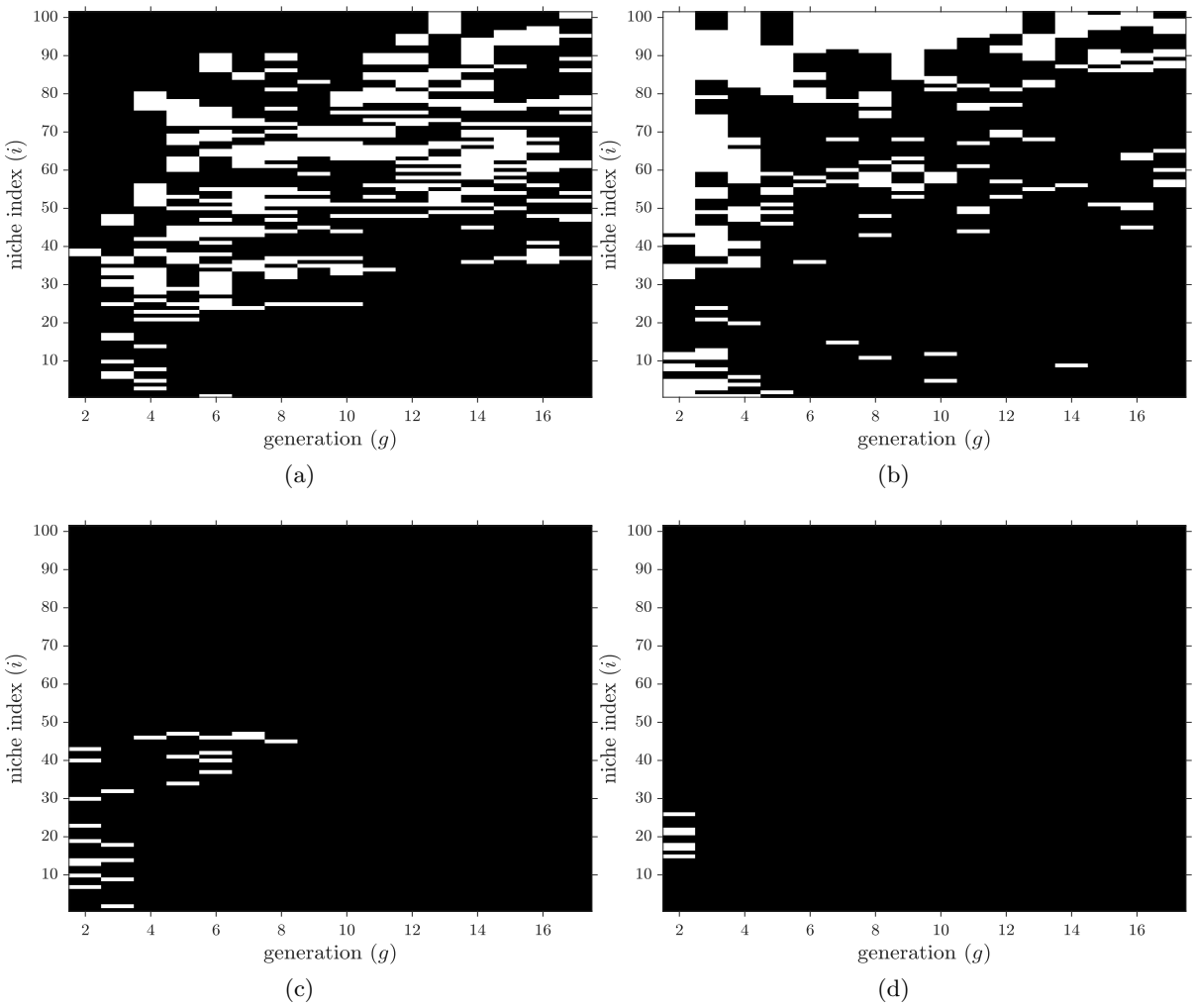


Figure E.1: Selection from offspring types during overlapping-niche descent. White indicates that $(\mathbf{p}_{g,i,1}, \mathbf{x}_{g,i,1})$, the individual in the (first) parent space of the i^{th} niche in generation g as described in Section C.1, was selected from a high momentum offspring in (a), a cross-niche offspring in (b), a sexual offspring in (c) and a random offspring in (d).

As is visible in Figure E.1, in early generations of overlapping-niche descent, sexual offspring and random offspring contribute to convergence in smaller values of λ while high momentum offspring and sexual offspring contribute to convergence broadly in λ, and in subsequent generations of overlapping-niche descent, high momentum offspring and sexual offspring contribute to convergence in larger values of λ.

To illustrate how cross-niche optimization contributes to convergence in overlapping-niche descent, I map selection from niches in Figure E.2.

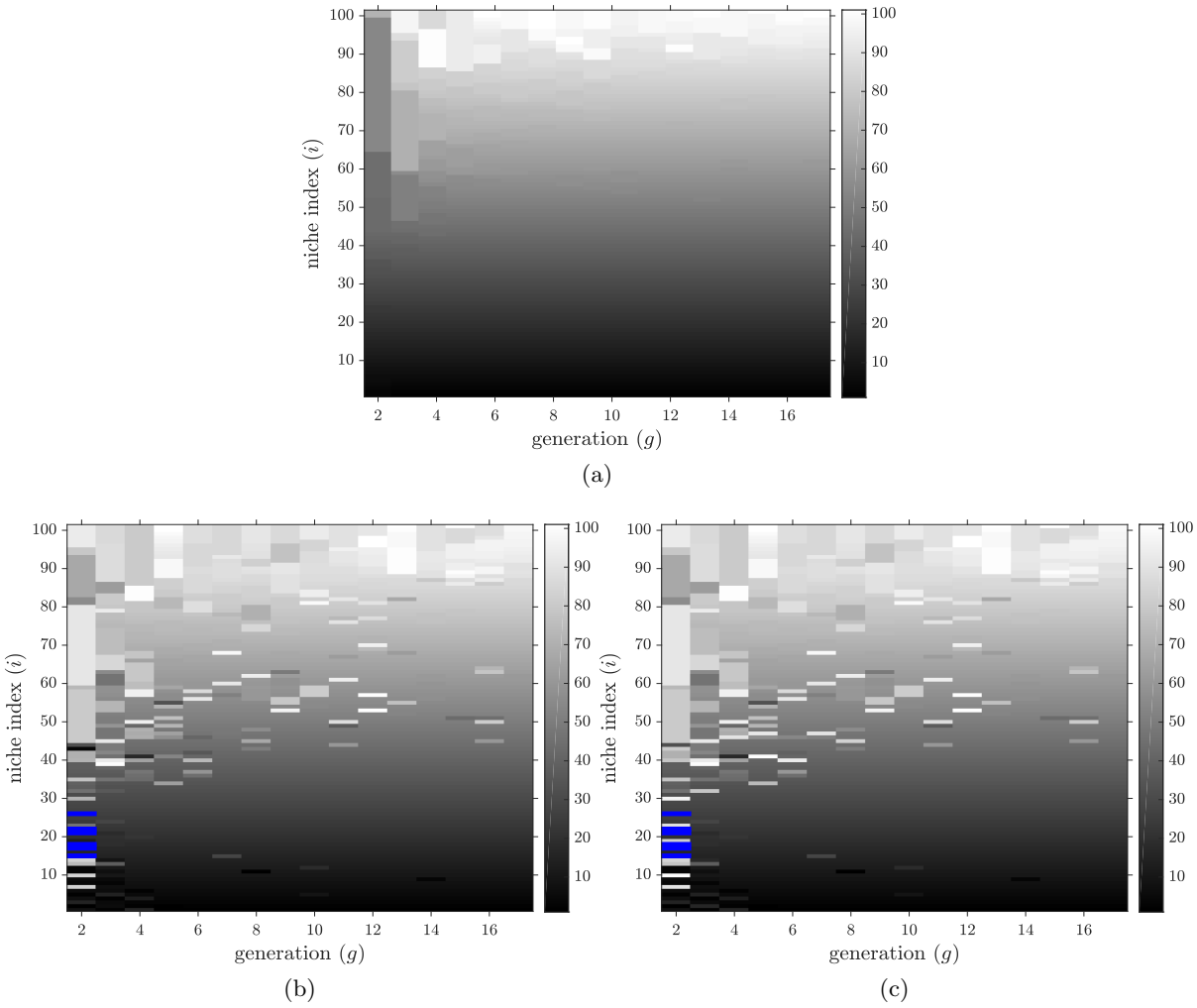


Figure E.2: Selection across niches during overlapping-niche descent. The individual in the (first) parent space of the i^{th} niche in generation g, $(\mathbf{p}_{g,i,1}, \mathbf{x}_{g,i,1})$, was selected from the niche shown by grayscale in (a) and was generated from parents(s) in the niche(s) shown by grayscale in (b) and (c). In (b) and (c), a blue value indicates that the selected individual was a random offspring and thus not generated from a parent in any niche.

As is visible in Figure E.2, cross-niche optimization ubiquitously contributes to convergence throughout overlapping-niche descent, and selection or generation from a parent from a niche with a larger value of λ contributes to convergence at least as much as selection or generation from a parent from a niche with a smaller value of λ. Interestingly, although $\tilde{r}(\lambda)$ converges more readily for smaller λ, selection or generation from a parent from a niche with a larger value of λ contributes to convergence in $\tilde{r}(\lambda)$ for smaller λ.

E.3.2 Prolongation in Overlapping-Niche Descent

As discussed in Section C.3, I prolong accelerated descent for an individual that appears to be converging to a value of $r(\mathbf{p}, \mathbf{x}; \lambda)$ below the least value of $r(\mathbf{p}, \mathbf{x}; \lambda)$ in its niche. To illustrate how

descent prolongation contributes to convergence in overlapping-niche descent, I map selection from individuals with prolonged accelerated descent in Figure E.3.

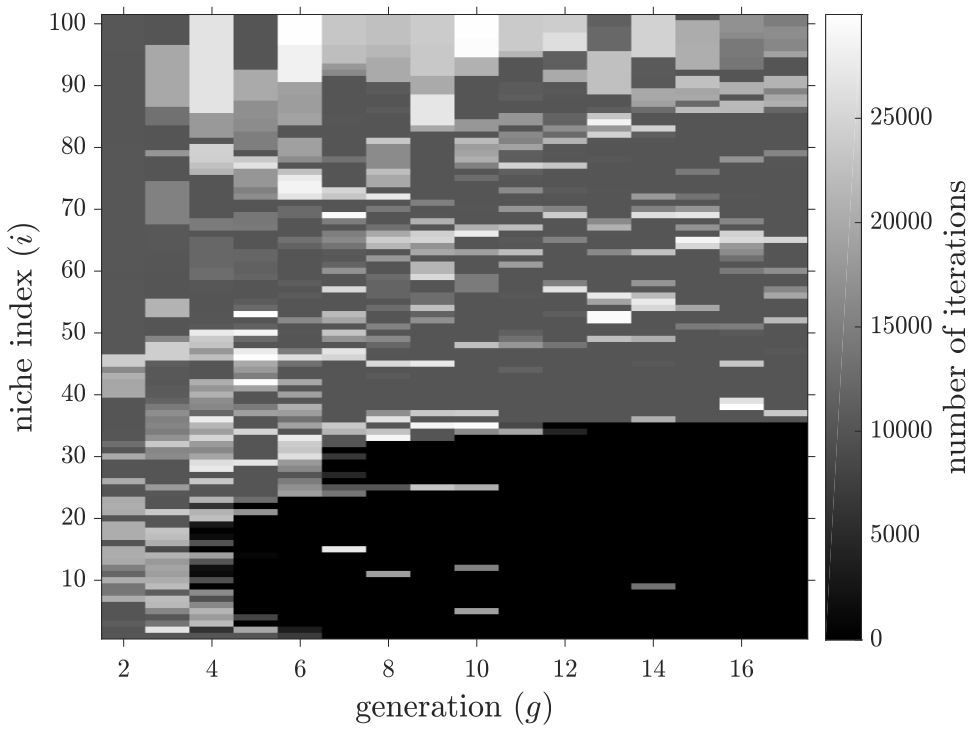


Figure E.3: Descent prolongation during overlapping-niche descent. The number of strict descent iterations are shown for the individual that was selected to occupy the (first) parent space of the i^{th} niche in generation g, $(\mathbf{p}_{g,i,1}, \mathbf{x}_{g,i,1})$. Descent prolongation occurs when the number of strict descent iterations exceeds $n_{\max} = 10^4$.

As is visible in Figure E.3, descent prolongation ubiquitously contributes to convergence throughout overlapping-niche descent.

E.3.3 Convergence During Accelerated Descent

As discussed in Section C.2, overlapping-niche descent includes accelerated descent, a variant of accelerated scaled gradient descent. Generally, accelerated descent follows a trajectory with rapid, sublinear convergence that develops into periods of superlinear convergence, which are punctuated by restarts. The number of strict descent iterations with sublinear convergence, the number of strict descent iterations with superlinear convergence, and convergence rates vary with initial parameters and state values and with λ. I show convergence plots that exemplify the convergence behavior of accelerated descent in Figure E.4.

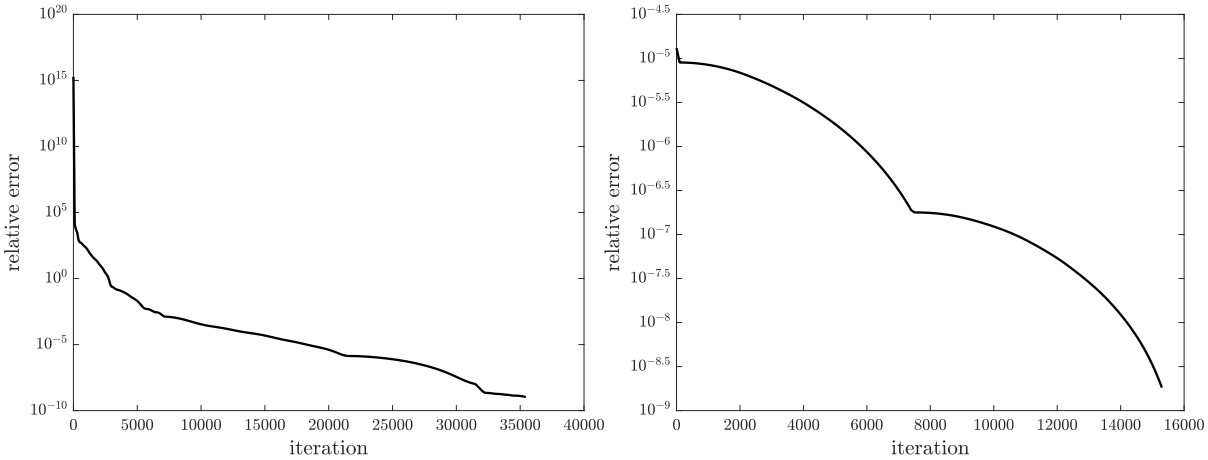


Figure E.4: Convergence behavior of accelerated descent. Rapid, sublinear convergence that develops into periods of superlinear convergence is shown in (a), for accelerated descent that begins with a random offspring and converges to $(\tilde{\mathbf{p}}^{\lambda_{22}}, \tilde{\mathbf{x}}^{\lambda_{22}})$. Periods of superlinear convergence that are punctuated by restarts is shown in (b), for accelerated descent that begins with a high momentum offspring and converges to $(\tilde{\mathbf{p}}^{\lambda_8}, \tilde{\mathbf{x}}^{\lambda_8})$. I calculate relative errors as $(r(\mathbf{p}, \mathbf{x}; \lambda) - r(\tilde{\mathbf{p}}^{\lambda}, \tilde{\mathbf{x}}^{\lambda}; \lambda))/(\tilde{\mathbf{p}}^{\lambda}, \tilde{\mathbf{x}}^{\lambda}; \lambda)$. Rather than misrepresenting relative errors near $(\tilde{\mathbf{p}}^{\lambda}, \tilde{\mathbf{x}}^{\lambda})$, I omit relative errors in the last 10^3 strict descent iterations of accelerated descent.

For reference, gradient descent and Nesterov's accelerated gradient method generally converge sublinearly, with respective upper bounds on errors of $O(n^{-1})$ and $O(n^{-2})$ for iteration number n [50], and Newton's method generally converges quadratically.

To demonstrate the importance of scaling in accelerated descent, as discussed in Section C.2.1, I apply accelerated descent without scaling, $s_{i,j} = 1$ for all $i \in \{1, 2, \ldots, n_v\}$ and $j \geq 1$ in equation (C.5), to the randomly generated individuals from the first generation of overlapping-niche descent and compare results from the optimization to analogous results from the optimization with accelerated descent. In doing so, for a balanced comparison, I only compare results for individuals with $n_{\max} = 10^4$ strict descent iterations during both accelerated descent and accelerated descent without scaling, where $n_{\max}$ is the maximum number of strict descent iterations described in Section C.2.2 and specified in Section E.1.5. Results are shown in Figure E.5.

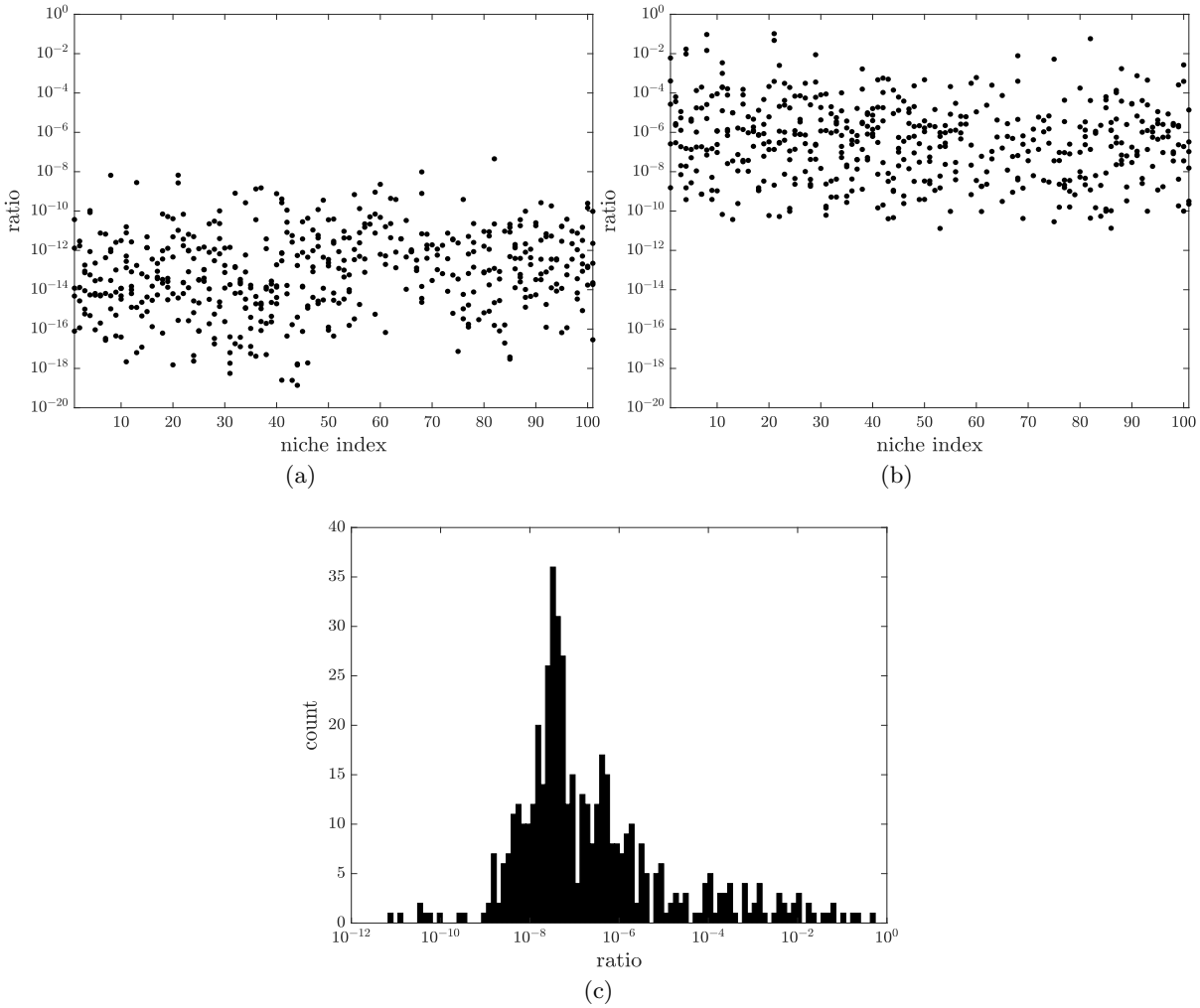


Figure E.5: A comparison of optimization using accelerated descent and accelerated descent without scaling. Final-to-initial ratios of $r(\mathbf{p}, \mathbf{x}; \lambda)$ are shown for accelerated descent in (a) and for accelerated descent without scaling in (b). A histogram of values for the ratio of the final value of $r(\mathbf{p}, \mathbf{x}; \lambda)$ from accelerated descent to the final value of $r(\mathbf{p}, \mathbf{x}; \lambda)$ from accelerated descent without scaling is shown in (c). In (a), (b), and (c), I show results for the randomly generated individuals in the first generation of overlapping-niche descent with $n_{\max} = 10^4$ strict descent iterations during both accelerated descent and accelerated descent without scaling and omit other results.

As is visible in Figure E.5, uniformly in all niches, accelerated descent dramatically outperforms accelerated descent without scaling. More precisely, in accordance with the histogram in panel (c) of Figure E.5, for the randomly generated individuals in the first generation of overlapping-niche descent (with $n_{\max} = 10^4$ strict descent iterations during both accelerated descent and accelerated descent without scaling), I find a median value for the ratio of the final value of $r(\mathbf{p}, \mathbf{x}; \lambda)$ from accelerated descent to the final value of $r(\mathbf{p}, \mathbf{x}; \lambda)$ from accelerated descent without scaling of $6.86 \cdot 10^{-8}$.

Appendix F

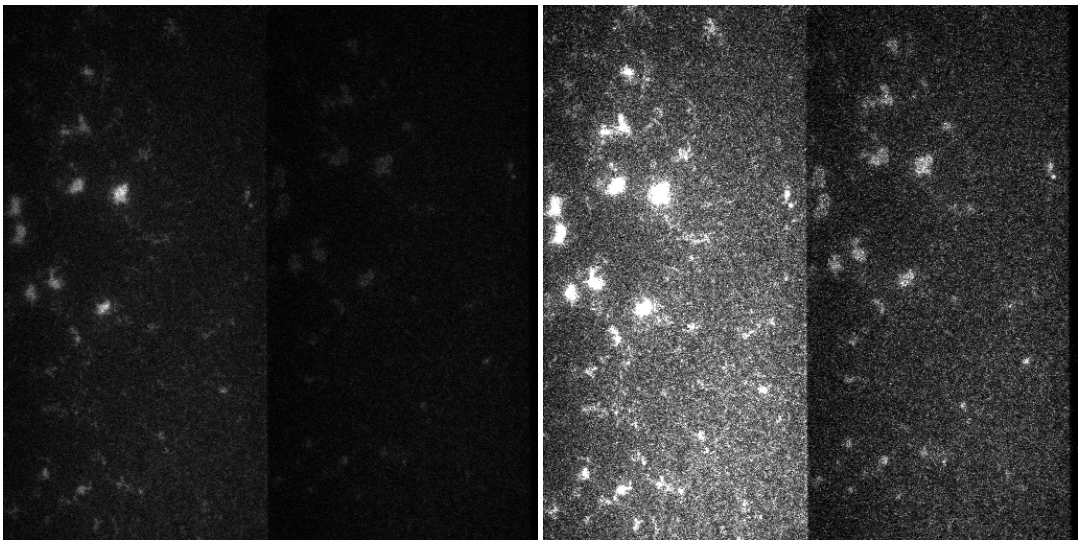
Extracting Near-Homogeneous Data

Here, I discuss details of data preprocessing and extracting spatially near-homogeneous time-course data from Ivanov and Mizuuchi's *in vitro* experimental measurements of the Min system [38].

F.1 Data Information

In the Ivanov and Mizuuchi experiments, in a 25 μm deep flow chamber, a buffer with 1.06 μM fluorescently labeled EFGP-MinD, 1.36 μM fluorescently labeled Alexa647-MinE, and 2.5 mM ATP was flowed at an average rate of 0.5 mm s^{-1} atop a supported lipid bilayer. On the supported lipid bilayer, densities of MinD and MinE, which were measured using total internal reflection microscopy (TIRF), oscillated near-homogeneously in space then formed into traveling waves. According to the Ivanov and Mizuuchi Supporting Information, rapid buffer flow compensated for local changes in the concentrations of reaction components in the buffer, as the buffer flow rate was three orders of magnitude faster than the typical traveling wave speed of MinD and MinE densities on the supported lipid bilayer and Taylor dispersion in the laminar flow was estimated to take place on a time scale that was about two orders of magnitude faster than the typical wave period of MinD and MinE densities on the supported lipid bilayer.

I analyze Ivanov and Mizuuchi's raw data, from the file **Movie1.stk**, using **MATLAB**, and the function **tiffread2** to import data. The data contains 2401 grayscale frames. Each frame is 512×512 pixels, and is a dual image, with the fluorescence intensity signals of EGFP-MinD on the left and the fluorescence intensity signals of Alexa647-MinE on the right. For visual clarification, an image of the 330^{th} data frame is shown in (a) of Figure F.1 and an enhanced (through the **MATLAB** function **imadjust**) image of the 330^{th} data frame is shown in (b) of Figure F.1.



(a) (b)

Figure F.1: The 330^{th} data frame as an image (a) and as an enhanced image (b).

F.2 Aligning Data Tracks

As is visible in (b) of Figure F.1, similarly shaped structures in MinD and MinE fluorescence intensity profiles seem to be aligned in rotation and scale, but not vertically. Also, MinD and MinE fluorescence intensity profiles require horizontal alignment because both signals are merged into a single image. I translationally align EGFP-MinD fluorescence intensity profiles and Alexa647-MinE fluorescence intensity profiles using structures in the 330^{th} data frame as location markers. I denote the intensity value of the i^{th} vertical pixel from the top and the j^{th} horizontal pixel from the left in the 330^{th} merged data frame by $m_{i,j}^{330}$, for $i \in \{1, 2, \ldots, 512\}$ and $j \in \{1, 2, \ldots, 512\}$. I select a square alignment preimage sufficiently within the interior of the MinE fluorescence intensity profile (shown in Figure F.2), which contains a unique arrangement of structures that are shaped similarly in both the MinD and MinE fluorescence intensity profiles.

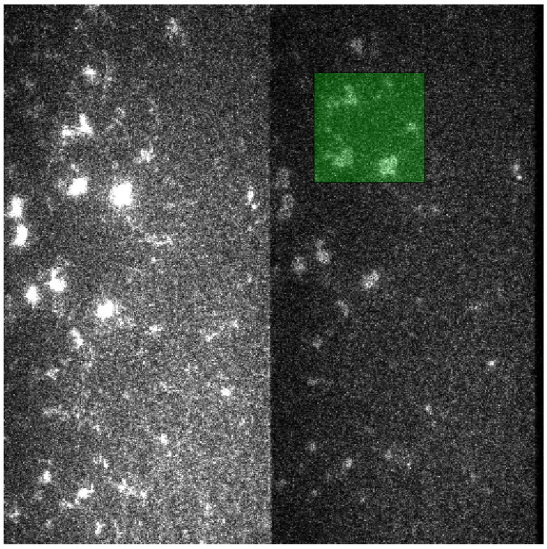


Figure F.2: Alignment preimage $[(p, q) = (65, 295), l = 106]$.

The alignment preimage has upper-left vertex $(p, q) = (65, 295)$ with side length $l = 106$ pixels. I compare the similarity between the alignment preimage in the MinE fluorescence intensity profile with an equally-sized alignment image in the MinD fluorescence intensity profile by normalized cross-correlation. For a square alignment image in the MinD fluorescence intensity profile with upper-left vertex (s, t) and side length l, the normalized cross-correlation between the alignment preimage and the alignment image is given by:

$$c(s,t) = \frac{\displaystyle\sum_{i=0}^{l-1}\sum_{j=0}^{l-1}(m^{330}_{p+i,q+j} - \mu_{p,q})(m^{330}_{s+i,t+j} - \mu_{s,t})}{\left(\displaystyle\sum_{i=0}^{l-1}\sum_{j=0}^{l-1}(m^{330}_{p+i,q+j} - \mu_{p,q})^2\right)^{\frac{1}{2}}\left(\displaystyle\sum_{i=0}^{l-1}\sum_{j=0}^{l-1}(m^{330}_{s+i,t+j} - \mu_{s,t})^2\right)^{\frac{1}{2}}}, \tag{F.1}$$

with alignment preimage and image mean values $\mu_{p,q}$ and $\mu_{s,t}$:

$$\mu_{p,q} = \frac{1}{l^2}\sum_{i=0}^{l-1}\sum_{j=0}^{l-1} m^{330}_{p+i,q+j}, \tag{F.2a}$$

$$\mu_{s,t} = \frac{1}{l^2}\sum_{i=0}^{l-1}\sum_{j=0}^{l-1} m^{330}_{s+i,t+j}. \tag{F.2b}$$

The MinD fluorescence intensity profile appears to end at the 252^{nd} horizontal pixel. Thus, for side length $l = 106$, square alignment images with upper-left vertices $(s, t) \in \{1, 2, \ldots, 407\} \times \{1, 2, \ldots, 147\}$ are contained entirely within the MinD fluorescence intensity profile. I calculate

$c(s,t)$ for all $(s,t) \in \{1,2,\ldots,407\} \times \{1,2,\ldots,147\}$; relative values are shown in Figure F.3.

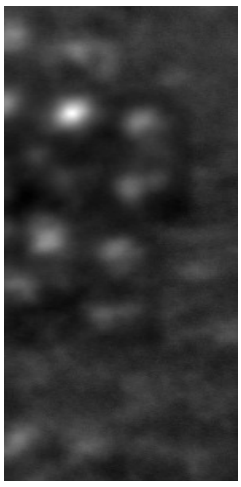


Figure F.3: Relative $c(s,t)$ values for $(s,t) \in \{1,2,\ldots,407\} \times \{1,2,\ldots,147\}$. Values increase with gradation from black to white.

For $(s,t) \in \{1,2,\ldots,407\} \times \{1,2,\ldots,147\}$, $c(s,t)$ has a maximum value of 0.56 at $(92,43)$. The alignment preimage in the MinE fluorescence intensity profile and the alignment image in the MinD fluorescence intensity profile with upper-left vertex $(s,t) = (92,43)$ are shown in Figure F.4.

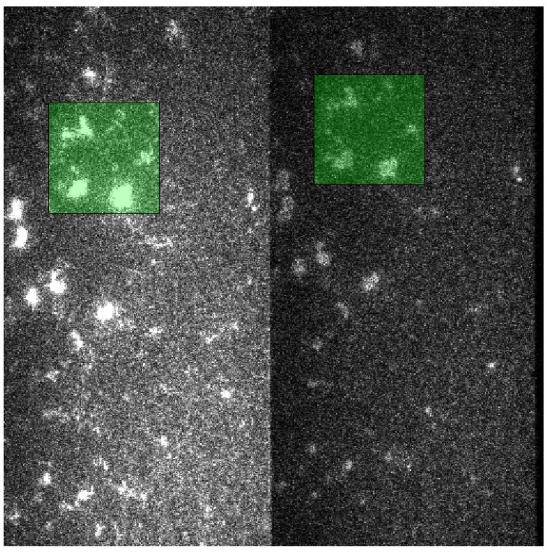


Figure F.4: Alignment preimage $[(p,q) = (65,295), l = 106]$ and alignment image $[(s,t) = (92,43), l = 106]$.

Thus, I align pixel $(65,295)$ in the MinE fluorescence intensity profile with pixel $(92,43)$ in the

MinD fluorescence intensity profile, a shift of $(27, -252)$ pixels. I apply the same shift to align all MinD and MinE fluorescence intensity data that has a one-to-one correspondence under the shift. Therefore, I assign value $m^{330}_{i+27,j}$ to aligned MinD fluorescence intensity data element $\ddot{d}^{330}_{i,j}$, and I assign value $m^{330}_{i,j+252}$ to aligned MinE fluorescence intensity data element $\ddot{e}^{330}_{i,j}$, for $i \in \{1, 2, \ldots, 485\}$ and $j \in \{1, 2, \ldots, 252\}$. I align MinD and MinE fluorescence intensity data identically in all data frames. During many data frames, MinD fluorescence intensity in the two right-most vertical pixels is significantly less than fluorescence intensity in neighboring pixels, and MinE fluorescence intensity in the three left-most vertical pixels is significantly more than fluorescence intensity in neighboring pixels. Thus, I remove the three left-most and two right-most vertical pixels from aligned MinD and MinE fluorescence intensity data. Therefore, for data frame t, I define aligned MinD and MinE fluorescence intensity data elements $\dot{d}^t_{i,j}$ and $\dot{e}^t_{i,j}$ to be data elements $\ddot{d}^t_{i,j+3} = m^t_{i+27,j+3}$ and $\ddot{e}^t_{i,j+3} = m^t_{i,j+252+3}$, for $i \in \{1, 2, \ldots, 485\}$, $j \in \{1, 2, \ldots, 247\}$, and $t \in \{1, 2, \ldots, 2401\}$. An enhanced image of aligned MinD and MinE fluorescence intensity data is shown in Figure F.5.

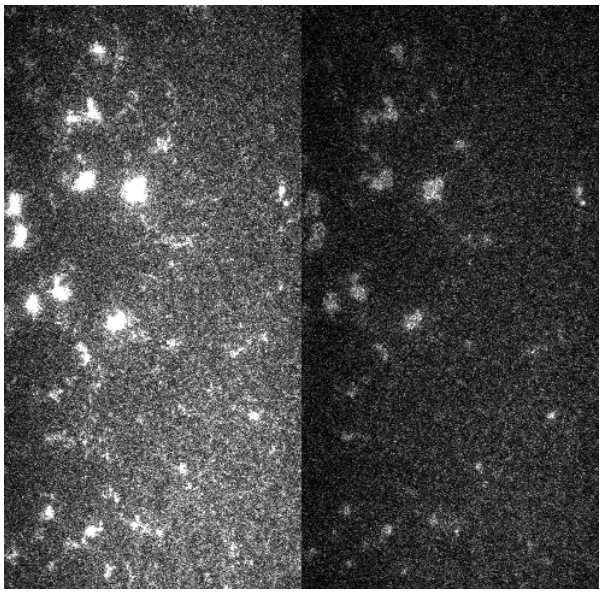


Figure F.5: Aligned data from the 330^{th} data frame as an image.

F.3 Preparing Aligned Data for Analysis

F.3.1 Temporal Partition of Data

Aligned MinD and MinE fluorescence intensity data contains the temporal evolution of six pulses, over roughly the entire spatial domain of the data, that slowly develop into persistent traveling wave fronts. I temporally partition aligned MinD and MinE fluorescence intensity

data, by temporal index, such that

$$\{1, 2, \ldots, 2401\} = \bigcup_{k=0}^{10} P_k =$$
$$\{1, \ldots, 224\} \cup \{225, \ldots, 343\} \cup \{344, \ldots, 546\} \cup \{547, \ldots, 719\} \cup$$
$$\{720, \ldots, 894\} \cup \{895, \ldots, 1064\} \cup \{1065, \ldots, 1259\} \cup \{1260, \ldots, 1388\} \cup$$
$$\{1389, \ldots, 1539\} \cup \{1540, \ldots, 1677\} \cup \{1678, \ldots, 2401\},$$

where P_0 contains temporal indices before the first global pulse in MinD and MinE fluorescence intensity; $P_1, P_2, \ldots, P_6$ each contain temporal indices of a single, roughly global MinD and MinE fluorescence intensity spike then fall; and P_7 contains the temporal indices of the beginning of MinD and MinE fluorescence intensity traveling wave formation, which further develops during the temporal indices of P_8 and P_9, and persists through the temporal indices of P_{10}. I denote aligned MinD and MinE fluorescence intensity data elements $d_{i,j}$ and $e_{i,j}$, in the l^{th} ordered index of the k^{th} temporal partition, as $d_{i,j}^{k,l}$ and $e_{i,j}^{k,l}$, for $i \in \{1, 2, \ldots, 485\}$, $j \in \{1, 2, \ldots, 247\}$, $k \in \{0, 1, \ldots, 10\}$, and $l \in \{1, 2, \ldots, n(P_k)\}$, where $n(P_k)$ denotes the cardinality of P_k.

F.3.2 Intensity Flattening

Bulk MinD and MinE, in the solution buffer, reach the middle of the flow cell, the site of fluorescence intensity measurements, during temporal partition P_0. Before bulk proteins reach the middle of the flow cell, MinD and MinE fluorescence intensities consist entirely of background noise. After bulk proteins reach the middle of the flow cell, MinD and MinE fluorescent intensities consist of background noise, bulk proteins, and some membrane-bound proteins. Mean MinD and MinE fluorescence intensity values over space, for each data frame in P_0, are shown in Figure F.6.

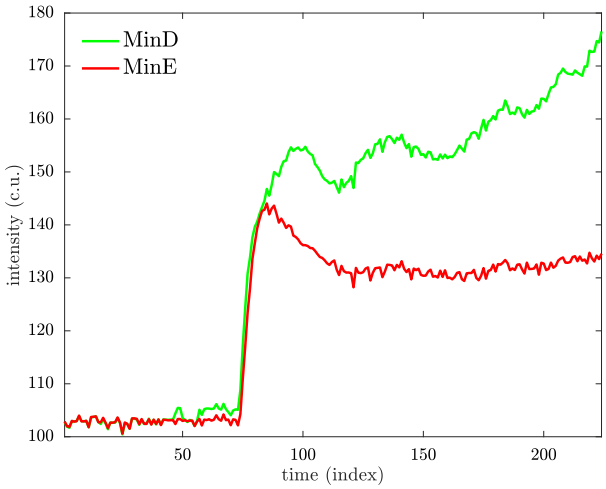
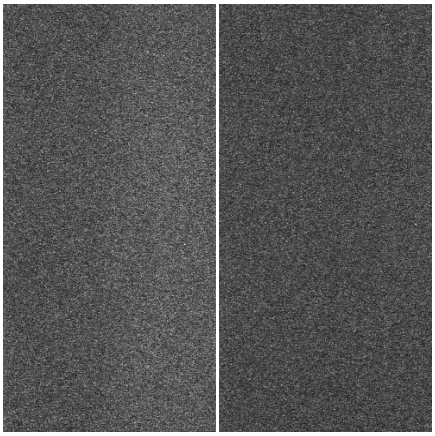


Figure F.6: Mean fluorescence intensities over space during temporal partition P_0. Intensity values are in camera units (c.u.).

As is visible in Figure F.6, MinD and MinE fluorescence intensities consist entirely of background noise from image 1 to image 56. After image 56, MinD fluorescence intensity increases slightly, then MinD and MinE fluorescence intensities increase dramatically, presumably when bulk proteins reach the middle of the flow cell. Mean MinD and MinE fluorescence intensities over time, from image 1 to image 56, are shown in Figure F.7.



(a) (b)

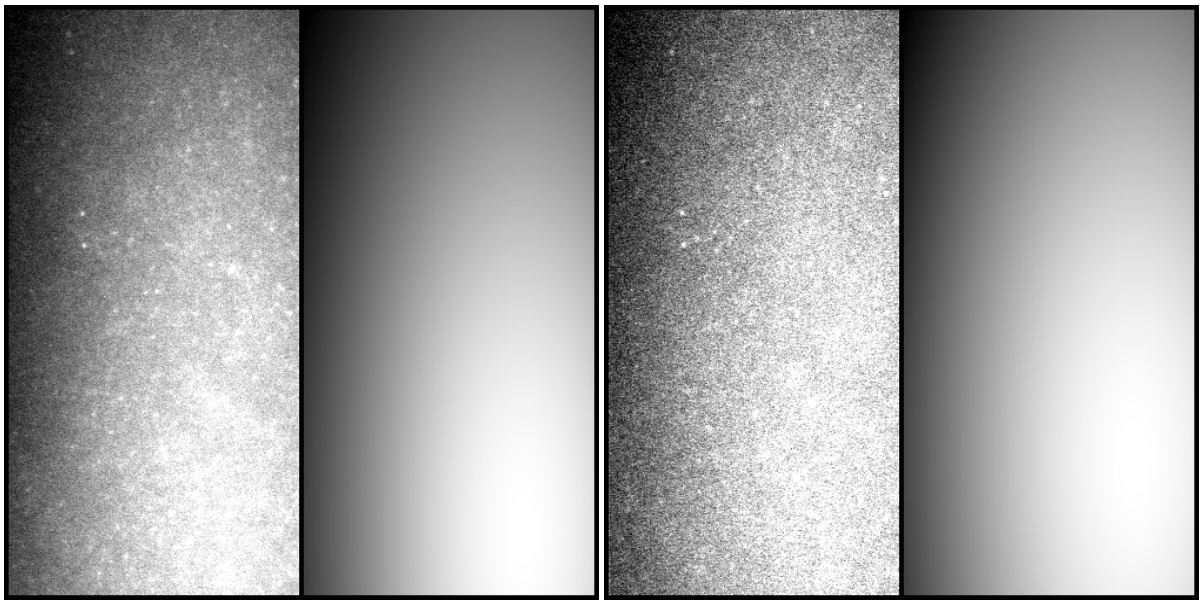
Figure F.7: Mean background intensities over time. MinD fluorescence intensity is shown in (a) and MinE fluorescence intensity is shown in (b).

As is visible in Figure F.7, background noise intensities are roughly uniform in space. MinD and MinE background fluorescence intensities, in all pixels, from image 1 to 56, have mean values $\mu_d^b = 102.9$ c.u. and $\mu_e^b = 102.8$ c.u. and standard deviations $\sigma_d^b = 12.3$ c.u. and $\sigma_e^b = 12.2$ c.u., where camera unit is abbreviated as c.u..

After bulk proteins reach the middle of the flow cell, MinD and MinE fluorescence intensities no longer appear roughly uniform in space. According to the Ivanov and Mizuuchi Supporting Information, "The illumination had a Gaussian shape in the field of view with a measured horizontal and vertical half maximum widths of 65×172 μm at 488 nm ", where 488 nm refers to the wavelength of MinD fluorescence excitation. The Gaussian spreads of MinD and MinE fluorescent intensities are visibly different. Thus, MinD and MinE fluorescence intensities require flattening to remove asymmetries in data acquisition. The Gaussian function, in two spatial dimensions x and y, has the form

$$g(x, y; A, x_0, y_0, k_x, k_y) = A \exp\Big(- \big(k_x(x - x_0)^2 + k_y(y - y_0)^2\big)\Big), \tag{F.3}$$

with scaling factor A, center point (x_0, y_0), and spread characterizing parameters k_x and k_y. The sum of Gaussian functions with identical center point and spread is a Gaussian function that retains the center point and spread. For static illumination sources, the Gaussian center points and spreads of MinD and MinE fluorescence intensity remain constant throughout data acquisition. During P_0, after bulk proteins reach the middle of the flow cell, apart from apparent Gaussian profiles, MinD and MinE fluorescence intensities appear to be roughly uniform in space. Thus, roughly, each MinD and MinE image contains spatially uniform background noise and a Gaussian intensity profile with static center point and spread. To generate MinD and MinE Gaussian profile data, initially, I subtract mean background intensity values, μ_d^b and μ_e^b, from MinD and MinE fluorescence intensity data, for image 85, when bulk proteins appear to have saturated the entire spatial domain of the data, through image 224, the final image in P_0. Then, for each image from 85 to 224, I normalize translated MinD and MinE data by the mean translated MinD and MinE value in the image. Finally, I generate MinD and MinE Gaussian profile data by calculating the mean normalized translated MinD and MinE value, from image 85 to 224, at each image pixel. Ultimately, generated MinD and MinE Gaussian data are roughly Gaussian, with mean values of 1. According to the Ivanov and Mizuuchi Supporting Information, the side length of each pixel is 6^{-1} μm. Thus, for a a horizontal half maximum width of 65 μm, $k_x = \ln(2) \cdot (6 \cdot 65 \text{ pixels})^{-2} \approx 4.56 \cdot 10^{-6}$ pixels^{-2}, and for a vertical half maximum widths of 172μm, $k_y = \ln(2) \cdot (6 \cdot 172 \text{ pixels})^{-2} \approx 6.51 \cdot 10^{-7}$ pixels^{-2}. Using the **MATLAB** function **lsqcurvefit**, I determine the parameters A, x_0, y_0, k_x, and k_y that minimize the sum of squared differences between the Gaussian function and generated MinD and MinE Gaussian profile data, with initial parameter estimates $A^0 = 1$, $k_x^0 = \ln(2) \cdot (6 \cdot 65 \text{ pixels})^{-2}$, $k_y^0 = \ln(2) \cdot (6 \cdot 172 \text{ pixels})^{-2}$, and $(x_0^0, y_0^0) = (123, 242)$, the center of the spatial domain. I compute MinD Gaussian parameters, $A^d = 1.43$, $k_x^d = 2.49 \cdot 10^{-5}$ pixels^{-2}, $k_y^d = 3.27 \cdot 10^{-6}$ pixels^{-2}, $x_0^d = 186.0$, and $y_0^d = 431.5$, and MinE Gaussian parameters, $A^e = 1.26$, $k_x^e = 1.50 \cdot 10^{-5}$ pixels^{-2}, $k_y^e = 2.53 \cdot 10^{-6}$ pixels^{-2}, $x_0^e = 196.3$, and $y_0^e = 371.2$. MinD and MinE Gaussian profile data and best fitting Gaussian functions are shown in Figure F.8.



(a) (b)

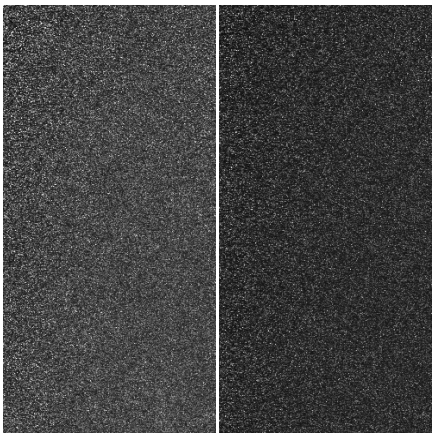
Figure F.8: Gaussian profile data and best fitting Gaussian functions. Generated Gaussian data is shown on the left and the best fitting Gaussian function is shown on the right, for MinD in (a) and MinE in (b).

Using fitted Gaussian parameters, I flatten all aligned MinD and MinE fluorescence intensity data to correct for Gaussian intensity profiles:

$$\bar{d}_{i,j}^{k,l} = \frac{A^d}{g\big(j, i; A^d, x_0^d, y_0^d, k_x^d, k_y^d\big)}\big(d_{i,j}^{k,l} - \mu_d^b\big), \tag{F.4}$$

$$\bar{e}_{i,j}^{k,l} = \frac{A^e}{g\big(j, i; A^e, x_0^e, y_0^e, k_x^e, k_y^e\big)}\big(e_{i,j}^{k,l} - \mu_e^b\big), \tag{F.5}$$

for $i \in \{1, 2, \ldots, 485\}$, $j \in \{1, 2, \ldots, 247\}$, $k \in \{0, 1, \ldots, 10\}$, and $l \in \{1, 2, \ldots, n(P_k)\}$. Confirming the assumption, during P_0, after bulk proteins reach the middle of the flow cell, flattened MinD and MinE fluorescence intensities appear to be roughly uniform in space. Flattened MinD and MinE fluorescence intensities are shown for image 224, the final image in P_0, in Figure F.9.



(a) (b)

Figure F.9: Flattened MinD and MinE fluorescence intensities for image 224. Flattened MinD fluorescence intensity is shown in (a) and flattened MinE fluorescence intensity is shown in (b).

F.3.3 Scaling Flattened Data

According to the Ivanov and Mizuuchi Supporting Information, by comparing the fluorescence intensity of EGFP-MinD with the fluorescence intensity of ParA-GFP, a protein with no affinity for the lipid bilayer, a EGFP-MinD fluorescence intensity of $7,000$ c.u. was found to correspond, approximately, to a surface density of $1.25 \cdot 10^4$ μm^{-2}. The Alexa647-MinE fluorescence intensity was not directly calibrated, but was estimated by direct comparison to experiments with similar dynamic outcomes involving MinE-EGFP and nonfluorescent MinD. Peak MinE-EGFP fluorescence intensities were found to be two to four times less than peak fluorescence intensities of EGFP-MinD, and the peak-to-peak ratio of 2.6 was used to normalize Alexa647-MinE data. I calibrate MinD and MinE fluorescence intensities from flattened fluorescence intensity data in temporal partition P_0.

MinD and MinE fluorescence intensities were measured using total internal reflection microscopy (TIRF). In TIRF, a light beam undergoes total internal reflection at the interface of a solution with a solid surface, producing an evanescent wave that excites fluorescent molecules near the solid surface [2]. The evanescent electric field intensity, I, decays with distance, z, from the solid surface:

$$I(z) = I_0 e^{-z/d}, \tag{F.6a}$$

$$d = \frac{\lambda_0}{4\pi}\left(n_1^2 \sin^2\theta - n_2^2\right)^{-\frac{1}{2}}, \tag{F.6b}$$

where I_0 is the electric field intensity at $z = 0$, λ_0 is the light wavelength in a vacuum, n_1 is the refractive index of the solid surface, n_2 is the refractive index of the solution, and θ is the angle of incidence [2]. According to the Ivanov and Mizuuchi Supporting Information, for MinD

fluorescence excitation, using a laser with a wavelength of $488 \cdot 10^{-3}$ μm, the penetration depth of the evanescent wave, d^d, was $128 \cdot 10^{-3}$ μm. The penetration depth of the evanescence wave was not explicitly characterized for MinE fluorescence excitation. From equation (F.6b) and d^d, for the experiment,

$$\frac{1}{4\pi}\left(n_1^2 \sin^2\theta - n_2^2\right)^{-\frac{1}{2}} = \frac{128}{488}. \tag{F.7}$$

MinE fluorophores were excited using a laser with a wavelength of $633 \cdot 10^{-3}$ μm. Thus, for MinE fluorescence excitation, the penetration depth of the evanescent wave, d^e, was $633 \cdot 10^{-3} \cdot 128 \cdot 488^{-1}$ μm $\approx 166 \cdot 10^{-3}$ μm.

Pixelated MinD and MinE fluorescence intensities, I^d and I^e, are the sums of bulk fluorescence intensities, I_D and I_E, and lipid bilayer-bound fluorescence intensities, I_d and I_e. I_D and I_E are proportional to the number of excited MinD and MinE fluorophores in solution, and I_d and I_e are proportional to the number of excited MinD and MinE fluorophores on the lipid bilayer; for MinD and MinE solution concentrations, c_D and c_E, and lipid bilayer-bound concentrations, c_d and c_e, in the 25 μm deep flow cell,

$$I^d = I_D + I_d : \begin{cases} I_D = \mu_d \cdot a \cdot \epsilon_d \cdot I_0^d \displaystyle\int_0^{25} c_D e^{-z/d^d} dz \\ \quad = \mu_d \cdot a \cdot \epsilon_d \cdot I_0^d \cdot c_D \cdot d^d (1 - 1.5 \cdot 10^{-85}) \\ \quad \approx \mu_d \cdot a \cdot \epsilon_d \cdot I_0^d \cdot c_D \cdot d^d \\ I_d = \mu_d \cdot a \cdot \epsilon_d \cdot I_0^d \cdot c_d, \end{cases} \tag{F.8a}$$

$$I^e = I_E + I_e : \begin{cases} I_E = \mu_e \cdot a \cdot \epsilon_e \cdot I_0^e \displaystyle\int_0^{25} c_E e^{-z/d^e} dz \\ \quad = \mu_e \cdot a \cdot \epsilon_e \cdot I_0^e \cdot c_E \cdot d^e (1 - 3.9 \cdot 10^{-66}) \\ \quad \approx \mu_e \cdot a \cdot \epsilon_e \cdot I_0^e \cdot c_E \cdot d^e \\ I_e = \mu_e \cdot a \cdot \epsilon_e \cdot I_0^e \cdot c_e, \end{cases} \tag{F.8b}$$

with MinD and MinE evanescent field intensities at $z = 0$, I_0^d and I_0^e, excitation scales, ϵ_d and ϵ_e, measurement scales, μ_d and μ_e, and pixel area, a. Thus, for camera unit (c.u.) to concentration (μm^{-2}) conversion factors, $\alpha_d = \left(\mu_d \cdot a \cdot \epsilon_d \cdot I_0^d\right)^{-1}$ and $\alpha_e = \left(\mu_e \cdot a \cdot \epsilon_e \cdot I_0^e\right)^{-1}$, to numerical precision,

$$c_D \cdot d^d = \alpha_d I_D, \tag{F.9a}$$

$$c_d = \alpha_d I_d, \tag{F.9b}$$

$$c_E \cdot d^e = \alpha_e I_E, \tag{F.9c}$$

$$c_e = \alpha_e I_e. \tag{F.9d}$$

Flowed bulk MinD and MinE protein concentrations are roughly constant in time. During P_0, in all data frames after bulk proteins reach the middle of the flow cell, flattened MinD and MinE fluorescent intensities appear roughly uniform in space, as shown in Figure F.9. Mean flattened MinD and MinE fluorescence intensity values over space, for each data frame in P_0, are shown in Figure F.10.

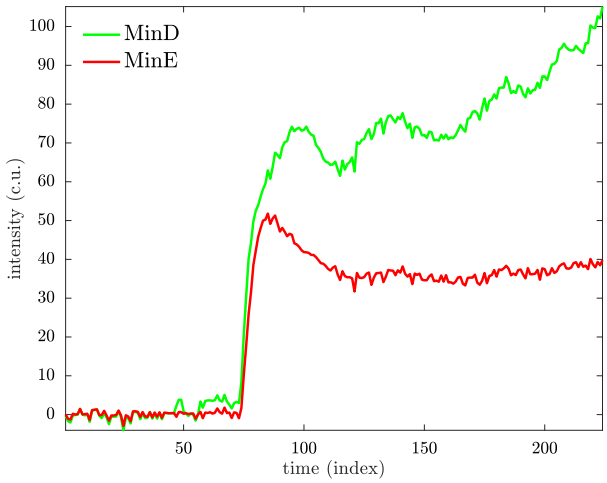


Figure F.10: Mean flattened fluorescence intensities over space during temporal partition P_0.

As is visible in Figure F.10, when bulk proteins reach the middle of the flow cell, MinD and MinE fluorescence intensities transiently peak, level off, then MinD fluorescence intensity gradually increases in time and MinE fluorescence intensity remains roughly constant in time. With no initial lipid bilayer-bound MinD and MinE and roughly constant bulk MinD and MinE concentrations, just after bulk proteins reach the middle of the flow cell, MinE fluorescence intensity, while roughly constant in time, consists almost entirely of bulk MinE protein fluorophore excitation. During a period with very little lipid bilayer-bound MinE and a roughly constant bulk MinD concentration, models of the evolution of lipid bilayer-bound MinD concentration, as defined in Equations (4.1), (4.6), (4.7), and (4.9) of Section 4.3, reduce to the form

$$\frac{dc_d}{dt} = (\omega_{D\to d} + \omega^d_{D\to d} c_d)(c_{\max} - c_d)/c_{\max} - \frac{\omega_{d\to D} c_s^{n_s} c_d}{c_s^{n_s} + c_d^{n_s}}, \tag{F.10}$$

where $c_{de}, c_e \approx 0$ and $\omega_{d\to D} = 0$ in equation (4.1), $c_{de}, c_e \approx 0$ and $n_s = 1$ in equation (4.6), $c_{de}, c_{ede}, c_e \approx 0$ and $n_s = 1$ in equation (4.7), and $c_{de}, c_{ded}, c_e \approx 0$ and $n_s = 1$ in equation (4.9). For small c_d, where c_d is very small compared to $c_{\max}$ and c_s, $(c_{\max} - c_d)/c_{\max} \approx 1$ and $c_s^{n_s}/(c_s^{n_s} + c_d^{n_s}) \approx 1$. Thus, models of the evolution of lipid bilayer-bound MinD concentration reduce to the form

$$\frac{dc_d}{dt} = \beta_1 + \beta_2 c_d, \tag{F.11}$$

where $\beta_1 = \omega_{D\to d}$ and $\beta_2 = \omega^d_{D\to d} - \omega_{d\to D}$, which has solution

$$c_d(t) = -\frac{\beta_1}{\beta_2} + \left(c_{d,0} + \frac{\beta_1}{\beta_2}\right)e^{\beta_2(t-t_0)}, \tag{F.12}$$

with concentration $c_{d,0}$ at time $t = t_0$. Converting to camera units, $I_d(t) = c_d(t)/\alpha_d$, equation (F.12) becomes

$$I_d(t) = -\frac{\gamma_1}{\beta_2} + \left(I_{d,0} + \frac{\gamma_1}{\beta_2}\right)e^{\beta_2(t-t_0)}, \tag{F.13}$$

where $\gamma_1 = \beta_1/\alpha_d$ and $I_{d,0} = c_{d,0}/\alpha_d$. Mean flattened MinE fluorescence intensity values over space appear to be roughly constant from image 122 through image 204. After image 204, mean flattened MinE fluorescence intensity values over space increase slightly. Thus, to decompose MinD fluorescence intensity, $I^d(t)$, which I approximate as the mean flattened MinD fluorescence intensity over space, into roughly constant bulk MinD fluorescence, $I_D(t)$, and lipid bilayer-bound MinD fluorescence, $I_d(t)$, I determine $I_d(t)$ characterizing parameters, γ_1, β_1, and $I_{d,0}$, that minimize the variance in MinD bulk fluorescence intensity, $I_D(t) = I^d(t) - I_d(t)$, from image 122 through image 204, using the **MATLAB** function **lsqcurvefit** with initial parameter estimates of 10^{-3}. I find that, for $t_0 = 122$ index, $\gamma_1 = 3.37 \cdot 10^{-2}$ c.u. index^{-1}, $\beta_1 = 3.36 \cdot 10^{-2}$ index^{-1}, and $I_{d,0} = 2.24 \cdot 10^{-1}$ c.u. minimize the variance in $I_D(t)$. The decomposition of $I^d(t)$ into $I_D(t)$ and $I_d(t)$ is shown in Figure (F.11).

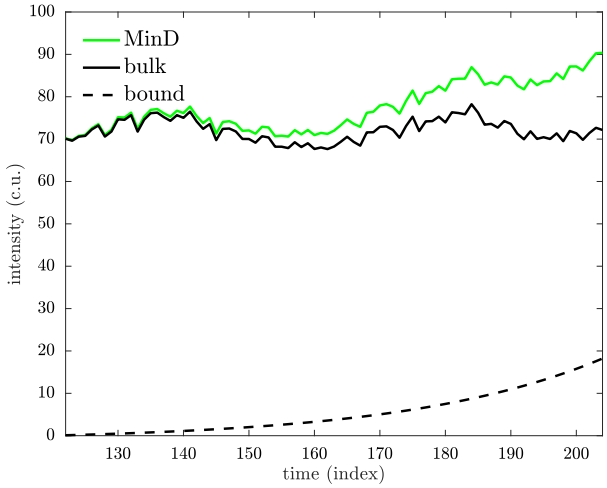


Figure F.11: The decomposition of MinD fluorescence intensity into bulk fluorescence intensity and lipid bilayer-bound fluorescence intensity.

Bulk MinD fluorescence intensity, $I_D(t)$, and bulk MinE fluorescence intensity, $I_E(t)$, which I approximate as the mean flattened MinE fluorescence intensity over space, from image 122 through image 204, are shown in Figure (F.12).

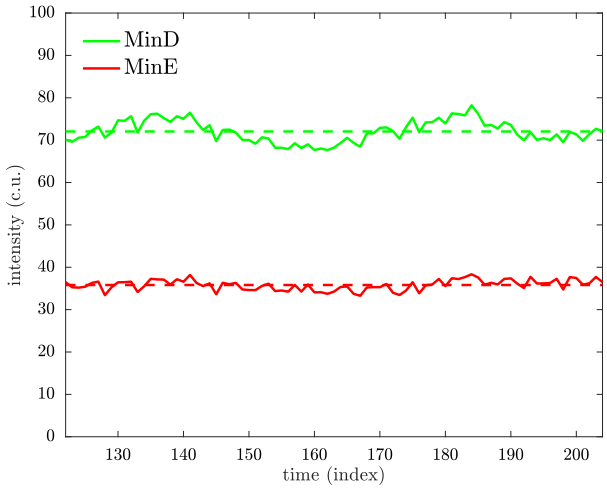


Figure F.12: Bulk MinD and MinE fluorescence intensities. Mean values are shown with dashed lines.

For camera unit to concentration conversion factors, α_d and α_e, from equation (F.9d),

$$\alpha_d = c_D \cdot d^d \cdot {I_D}^{-1}, \tag{F.14a}$$

$$\alpha_e = c_E \cdot d^e \cdot {I_E}^{-1}. \tag{F.14b}$$

${I_D}^{-1}$ and ${I_E}^{-1}$ have mean values $1.392 \cdot 10^{-2}$ c.u. and $2.796 \cdot 10^{-2}$ c.u. and standard deviations $4.9 \cdot 10^{-4}$ c.u. and $9.8 \cdot 10^{-4}$ c.u.. The flowed buffer contains 1.06 μM $= 6.383 \cdot 10^2$ μm^{-3} MinD and 1.36 μM $= 8.190 \cdot 10^2$ μm^{-3} MinE. Thus, α_d and α_e have mean values 1.137 μm^{-2} c.u.$^{-1}$ and 3.802 μm^{-2} c.u.$^{-1}$ and standard deviations 0.040 μm^{-2} c.u.$^{-1}$ and 0.133 μm^{-2} c.u.$^{-1}$. I approximate α_d and α_e by mean values, $\alpha_d = 1.137$ μm^{-2} c.u.$^{-1}$ and $\alpha_e = 3.802$ μm^{-2} c.u.$^{-1}$. Comparatively, Ivanov and Mizuuchi measured α_d, approximately, as 1.786 μm^{-2} c.u.$^{-1}$. I scale flattened MinD and MinE fluorescence intensity data to generate MinD and MinE density data:

$$d_{i,j}^{k,l} = \alpha_d \cdot \bar{d}_{i,j}^{k,l}, \tag{F.15}$$

$$e_{i,j}^{k,l} = \alpha_e \cdot \bar{e}_{i,j}^{k,l}, \tag{F.16}$$

for $i \in \{1, 2, \ldots, 485\}$, $j \in \{1, 2, \ldots, 247\}$, $k \in \{0, 1, \ldots, 10\}$, and $l \in \{1, 2, \ldots, n(P_k)\}$.

F.4 Finding Spatially Homogeneous Data

F.4.1 Spatially Near-Homogeneous Model Reductions

A partial differential equation description of some dynamic process with m states, $u_1, u_2, \ldots, u_m$, in n-dimensional space, $\mathbf{x} = x_1, x_2, \ldots, x_n$, and time, t, is characterized by the system of

equations:

$$\frac{\partial u_i(\mathbf{x},t)}{\partial t} = f_i\left(\mathbf{x}, t, u_1, \ldots, u_m, \frac{\partial u_1}{\partial x_1}, \ldots, \frac{\partial u_1}{\partial x_n}, \frac{\partial^2 u_1}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 u_1}{\partial x_1 \partial x_n}, \ldots, \frac{\partial u_2}{\partial x_1}, \ldots\right), \tag{F.17}$$

for some functions f_i and $i \in \{1, 2, \ldots, m\}$. If $u_1, u_2, \ldots, u_m$ evolve near-homogeneously over some spatial domain, Ω, then

$$u_i(\mathbf{x},t) = \mu_i(t) + \varepsilon g_i(\mathbf{x},t), \tag{F.18}$$

with spatially homogeneous state-evolution $\mu_i(t)$, small scaling factor $\varepsilon \geq 0$, and local state-variation $g_i(\mathbf{x},t)$, for $\mathbf{x} \in \Omega$ and $i \in \{1, 2, \ldots, m\}$. As $\varepsilon \to 0^+$, $u_i(\mathbf{x},t) \to \mu_i(t)$, and thus all partial derivatives of $u_1, u_2, \ldots, u_m$ with respect to $x_1, x_2, \ldots, x_n$ converge to a value of zero. Hence,

$$\begin{aligned}\lim_{\varepsilon \to 0^+} \frac{\partial u_i(\mathbf{x},t)}{\partial t} = \\ \lim_{\varepsilon \to 0^+} f_i\left(\mathbf{x}, t, u_1, \ldots, u_m, \frac{\partial u_1}{\partial x_1}, \ldots, \frac{\partial u_1}{\partial x_n}, \frac{\partial^2 u_1}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 u_1}{\partial x_1 \partial x_n}, \ldots, \frac{\partial u_2}{\partial x_1}, \ldots\right) = \\ f_i\left(\mathbf{x}, t, \mu_1, \ldots, \mu_m, 0, \ldots, 0, 0, \ldots, 0, \ldots, 0, \ldots\right) = f_i^{\mu}\left(t, \mu_1, \ldots, \mu_m\right),\end{aligned} \tag{F.19}$$

for $\mathbf{x} \in \Omega$ and $i \in \{1, 2, \ldots, m\}$. For f_i analytic in ε,

$$\begin{aligned}f_i\left(\mathbf{x}, t, u_1, \ldots, u_m, \frac{\partial u_1}{\partial x_1}, \ldots, \frac{\partial u_1}{\partial x_n}, \frac{\partial^2 u_1}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 u_1}{\partial x_1 \partial x_n}, \ldots, \frac{\partial u_2}{\partial x_1}, \ldots\right) = \\ f_i\left(\mathbf{x}, t, \mu_1 + \varepsilon g_1, \ldots, \mu_m + \varepsilon g_n, \varepsilon \frac{\partial u_1}{\partial x_1}, \ldots, \varepsilon \frac{\partial g_1}{\partial x_n}, \varepsilon \frac{\partial^2 g_1}{\partial x_1 \partial x_1}, \ldots, \varepsilon \frac{\partial^2 g_1}{\partial x_1 \partial x_n}, \ldots\right) = \\ f_i^{\mu}\left(t, \mu_1, \ldots, \mu_m\right) + O(\varepsilon),\end{aligned} \tag{F.20}$$

for $\mathbf{x} \in \Omega$ and $i \in \{1, 2, \ldots, m\}$. Thus,

$$\frac{\partial u_i(\mathbf{x},t)}{\partial t} = \frac{d\mu_i(t)}{dt} + \varepsilon \frac{\partial g_i(\mathbf{x},t)}{\partial t} = f_i^{\mu}\left(t, \mu_1, \ldots, \mu_m\right) + O(\varepsilon), \tag{F.21}$$

for $\mathbf{x} \in \Omega$ and $i \in \{1, 2, \ldots, m\}$. Hence, to leading order,

$$\frac{d\mu_i(t)}{dt} = f_i^{\mu}\left(t, \mu_1, \ldots, \mu_m\right) \tag{F.22}$$

for $\mathbf{x} \in \Omega$ and $i \in \{1, 2, \ldots, m\}$. Therefore, the global behavior of a near-homogeneous process is described to leading order by a system of ordinary differential equations, the spatially-homogeneous reduction of the system's partial differential equation description.

F.4.2 Finding Spatially Near-Homogeneous Data

I assume that near-homogeneous MinD and MinE densities at consecutive measurements result from near-homogeneous MinD and MinE density evolutions. MinD and MinE densities are linear combinations of unobserved state densities; I assume that near-homogenous MinD and MinE densities result from linear combinations of near-homogeneous unobserved state densities. Therefore, as discussed in Section F.4.1, I model sequentially near-homogeneous MinD and MinE densities by spatially-homogeneous partial differential equation model reductions.

Classifying near-homogeneous MinD and MinE densities requires a measure of data homogeneity, or, alternatively, a measure of data inhomogeneity. I measure data relative inhomogeneity, h, by the ratio of the data standard deviation to the data mean, the ratio of data variation to data uniformity. For strictly positive data, $h \geq 0$, and $h = 0$ if and only if data is constant. Thus, h is a well defined measure of MinD and MinE density inhomogeneity. For density data $c^{k,l}$, from the l^{th} data frame of the k^{th} temporal partition, I calculate the relative inhomogeneity of $c^{k,l}$ over $D(i,j,r)$, a disk with center at the middle of the $(i,j)^{\text{th}}$ pixel and radius r:

$$h^r \circ c^{k,l}_{i,j} = h\big(c^{k,l}|_{D(i,j,r)}\big) = \\ \big(\mu^r \circ c^{k,l}_{i,j}\big)^{-1} \left(\sum_{m=-r}^{r} \sum_{n=-r}^{r} \frac{\rho_{i+m,j+n}}{\pi r^2} \Big(c^{k,l}_{i+m,j+n} - \mu^r \circ c^{k,l}_{i,j} \Big)^2 \right)^{\frac{1}{2}}, \tag{F.23a}$$

with mean value of $c^{k,l}$ over $D(i,j,r)$,

$$\mu^r \circ c^{k,l}_{i,j} = \mu\big(c^{k,l}|_{D(i,j,r)}\big) = \sum_{m=-r}^{r} \sum_{n=-r}^{r} \frac{\rho_{i+m,j+n}}{\pi r^2} c^{k,l}_{i+m,j+n}, \tag{F.23b}$$

where $c^{k,l}_{i+m,j+n}$ is the value of $c^{k,l}$ at the $(i+m,j+n)^{\text{th}}$ pixel, and $\rho_{i+m,j+n}$ is the fraction of the $(i+m,j+n)^{\text{th}}$ pixel contained within $D(i,j,r)$.

MinD and MinE densities appear to globally vary over areas of hundreds to hundreds of thousands of pixels, with local pixel-to-pixel variation. Local pixel-to-pixel variation increases relative inhomogeneity, obscuring measurements of underlying global density variation. Surfaces are locally well approximated by tangent planes. Thus, I locally fit MinD and MinE densities by planes to approximate global MinD and MinE density surfaces. For density data $c^{k,l}$, I calculate the plane over a disk $D(i,j,r)$ that fits $c^{k,l}$ best in the least squares sense, and determine the value of the plane at the middle of the $(i,j)^{\text{th}}$ pixel:

$$p\big(c^{k,l}|_{D(i,j,r)}\big) = \hat{\beta}_x(x-i) + \hat{\beta}_y(y-j) + \hat{\beta}_0, \tag{F.24a}$$

$$(\hat{\beta}_x, \hat{\beta}_y, \hat{\beta}_0) = \arg \min_{(\beta_x,\beta_y,\beta_0)} \sum_{m=-r}^{r} \sum_{n=-r}^{r} \rho_{i+m,j+n} \big(\beta_x m + \beta_y n + \beta_0 - c^{k,l}_{i+m,j+n}\big)^2, \tag{F.24b}$$

$$g^r \circ c^{k,l}_{i,j} = g\big(c^{k,l}|_{D(i,j,r)}\big) = p\big(c^{k,l}|_{D(i,j,r)}\big)\Big|_{(x,y)=(i,j)} = \hat{\beta}_0, \tag{F.24c}$$

where $\rho_{i+m,j+n}$ is the fraction of the $(i+m, j+n)^{\text{th}}$ pixel contained within $D(i, j, r)$; I determine $\hat{\beta}_y, \hat{\beta}_x, \hat{\beta}_0$ by solving the weighted normal equations.

I choose the local density planar-fitting disk radius, $\mathring{r}$, such that $D(i, j, \mathring{r})$ covers 100 square pixels, the lower end of the visible MinD and MinE global density variation scale. I choose the relative inhomogeneity disk radius, $\bar{r}$, such that $D(i, j, \bar{r})$ covers 1000 square pixels, an order of magnitude more pixels than $D(i, j, \mathring{r})$ covers. Thus, $\mathring{r} = (100/\pi)^{1/2}$ pixels ≈ 5.6 pixels and $\bar{r} = (1000/\pi)^{1/2}$ pixels ≈ 17.8 pixels. In an experiment similar to that of Ivanov and Mizuuchi, Loose *et al* measured the motilities of MinD and MinE in traveling protein waves on the lipid bilayer. They found that motilities varied along the wave, but were well characterized by diffusion in sections of the wave, with maximal MinD and MinE diffusion coefficients of 0.374 ± 0.022 μm^2 s^{-1} and 0.320 ± 0.023 μm^2 s^{-1} [44]. Thus, maximal MinD and MinE root mean squared displacements, over the time between measurements, 3 s, are approximately $(4 \cdot 0.374$ μm^2 $\text{s}^{-1} \cdot 3$ s$)^{1/2} \cdot 6$ pixels $\mu\text{m}^{-1} \approx 12.7$ pixels and $(4 \cdot 0.320$ μm^2 $\text{s}^{-1} \cdot 3$ s$)^{1/2} \cdot 6$ pixels $\mu\text{m}^{-1} \approx 11.8$ pixels. Therefore, $\bar{r}$ is on the scale of MinD and MinE mobilities between measurements. I calculate the relative inhomogeneity of planar-fit MinD and MinE density data:

$$h^{\bar{r}} \circ g^{\mathring{r}} \circ d_{i,j}^{k,l} \text{ and } h^{\bar{r}} \circ g^{\mathring{r}} \circ e_{i,j}^{k,l}, \tag{F.25}$$

for $i \in \{\bar{R}+1, \bar{R}+2, \ldots, 485-\bar{R}\}$ and $j \in \{\bar{R}+1, \bar{R}+2, \ldots, 247-\bar{R}\}$, where $\bar{R} = \lceil \bar{r} + \mathring{r} - 1/2 \rceil$, the ceiling of $\bar{r} + \mathring{r} - 1/2$, and for $k \in \{0, 1, \ldots, 10\}$ and $l \in \{1, 2, \ldots, n(P_k)\}$.

During temporal partition P_0, MinD and MinE densities consist mainly of bulk proteins in the well-mixed, flowed solution buffer. Thus, to determine basal relative inhomogeneity scales, I calculate mean relative inhomogeneity values of planar-fit MinD and MinE density data, centered at mid data pixel $(243, 124)$, from image 122 through image 204 of temporal partition P_0, the images of bulk fluorescence intensity estimation (as discussed in Section F.3.3); relative inhomogeneity values are shown in Figure F.13; I find mean MinD and MinE relative inhomogeneity values $\mu_d^h = 0.0837$ and $\mu_e^h = 0.1148$, with standard deviations of $\sigma_d^h = 0.0129$ and $\sigma_e^h = 0.0188$.

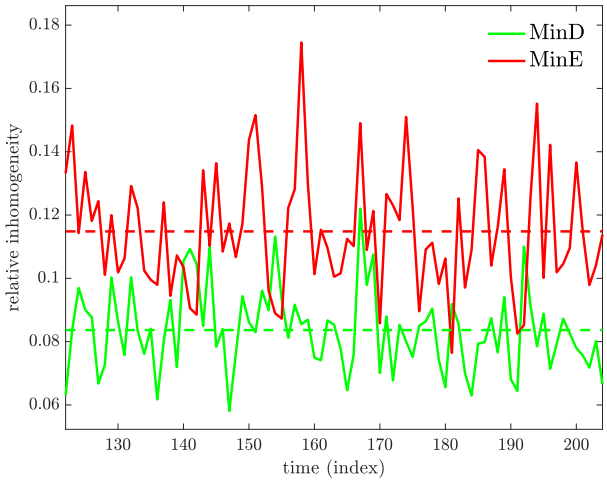
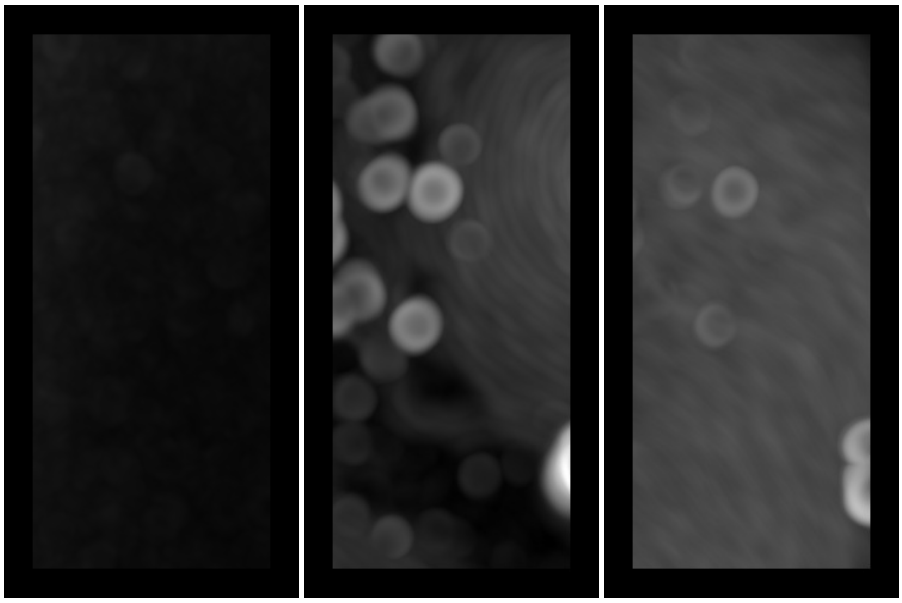


Figure F.13: Relative inhomogeneity values of planar-fit MinD and MinE density data in temporal partition P_0, $h^{\bar{r}} \circ g^{\mathring{r}} \circ d^{k,l}_{i,j}$ and $h^{\bar{r}} \circ g^{\mathring{r}} \circ e^{k,l}_{i,j}$ for $(i,j) = (243, 124)$, $k = 0$, and $l \in \{122, 123, \ldots, 204\}$. Mean values are shown with dashed lines.

To find planar-fit MinD and MinE density data that is uniformly homogenous, on basal relative inhomogeneity scales, throughout a temporal partition, I measure the scaled sum of maximal MinD and MinE relative inhomogeneity values in each temporal partition:

$$H^k_{i,j} = \frac{\max\left\{h^{\bar{r}} \circ g^{\mathring{r}} \circ d^{k,l}_{i,j} : l \in \{1, \ldots, n(P_k)\}\right\}}{\mu^h_d} + \frac{\max\left\{h^{\bar{r}} \circ g^{\mathring{r}} \circ e^{k,l}_{i,j} : l \in \{1, \ldots, n(P_k)\}\right\}}{\mu^h_e}, \tag{F.26}$$

for $i \in \{\bar{R}+1, \bar{R}+2, \ldots, 485-\bar{R}\}$, $j \in \{\bar{R}+1, \bar{R}+2, \ldots, 247-\bar{R}\}$, and $k \in \{0, 1, \ldots, 10\}$. Values of $H^k_{i,j}$ are shown in Figure F.14 for temporal partitions P_0 (images 122 through 204), P_5, and P_9.



(a) (b) (c)

Figure F.14: Scaled sums of maximal MinD and MinE relative inhomogeneity values in temporal partitions P_0 (images 122 through 204), P_5, and P_9. $H^k_{i,j}$ are shown for $k = 0$ in (a), $k = 5$ in (b), and $k = 9$ in (c), for $i \in \{\bar{R}+1, \bar{R}+2, \ldots, 485-\bar{R}\}$ and $j \in \{\bar{R}+1, \bar{R}+2, \ldots, 247-\bar{R}\}$. Values increase with gradation from black, with a value of 1.89, to white, with a value of 24.1.

As is visible in Figure F.14: in temporal partition P_0, consisting mainly of bulk flow, planar-fit MinD and MinE densities are uniformly homogeneous at all spatial location; in temporal partition P_5, during a MinD and MinE density pulse, planar-fit MinD and MinE densities are uniformly homogeneous at some spatial locations; and in temporal partition P_9, during MinD and MinE density traveling waves, planar-fit MinD and MinE densities are uniformly inhomogeneous at all spatial locations.

The minimum value of $H^k_{i,j}$ occurs at the $(436, 204)^{\text{th}}$ pixel of temporal partition P_5, $H^5_{436,204} = 1.89$. Relative inhomogeneity values of planar-fit MinD and MinE density data, at pixel $(436, 204)$, during temporal partition P_5, are shown in Figure F.15. For comparison, relative inhomogeneity values of planar-fit MinD and MinE density data, at mid data pixel $(243, 124)$, during temporal partition P_9, are also shown in Figure F.15.

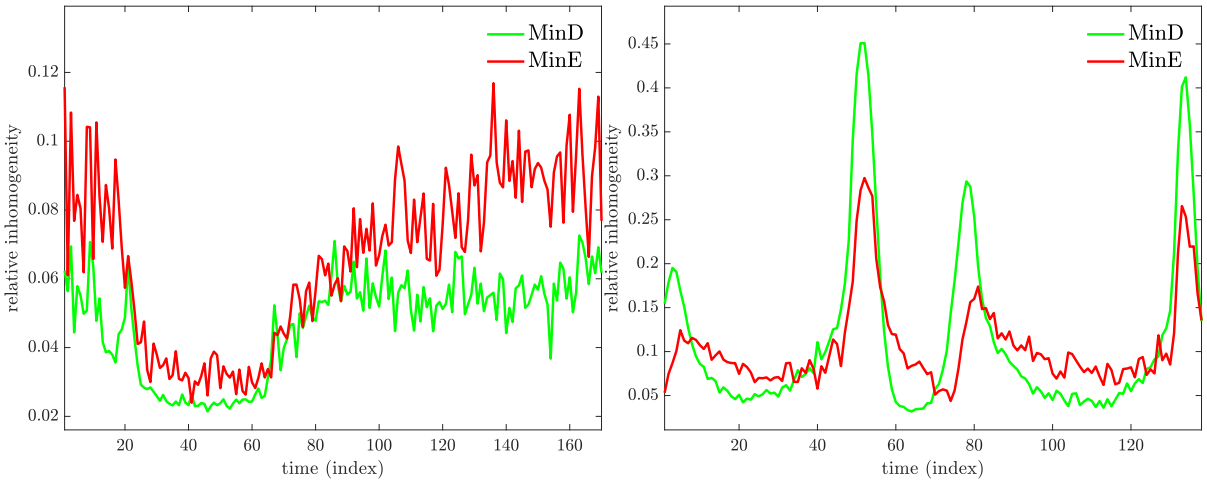


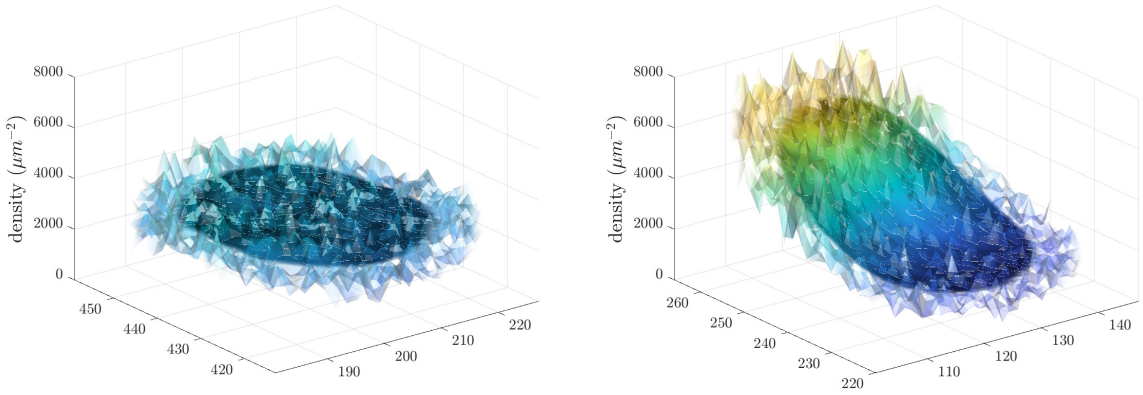
Figure F.15: Relative inhomogeneity values of planar-fit MinD and MinE density data. $h^{\bar{r}} \circ g^{\mathring{r}} \circ d^{k,l}_{i,j}$ and $h^{\bar{r}} \circ g^{\mathring{r}} \circ e^{k,l}_{i,j}$ are shown, for $(i,j) = (436, 204)$, $k = 5$, and $l \in \{1, 2, \ldots, 170\}$, in (a), and for $(i,j) = (243, 124)$, $k = 9$, and $l \in \{1, 2, \ldots, 138\}$, in (b).

Comparatively,

$$\begin{cases} \max\left\{h^{\bar{r}} \circ g^{\mathring{r}} \circ d^{5,l}_{436,204} : l \in \{1, 2, \ldots, n(P_5)\}\right\} = 0.87 \cdot \mu^h_d \\ \max\left\{h^{\bar{r}} \circ g^{\mathring{r}} \circ e^{5,l}_{436,204} : l \in \{1, 2, \ldots, n(P_5)\}\right\} = 1.02 \cdot \mu^h_e, \end{cases} \tag{F.27a}$$

$$\begin{cases} \max\left\{h^{\bar{r}} \circ g^{\mathring{r}} \circ d^{9,l}_{243,124} : l \in \{1, 2, \ldots, n(P_9)\}\right\} = 5.39 \cdot \mu^h_d \\ \max\left\{h^{\bar{r}} \circ g^{\mathring{r}} \circ e^{9,l}_{243,124} : l \in \{1, 2, \ldots, n(P_9)\}\right\} = 2.59 \cdot \mu^h_e. \end{cases} \tag{F.27b}$$

Thus, relative inhomogeneity values of planar-fit MinD and MinE density data, centered at pixel $(436, 204)$, during temporal partition P_5, do not significantly exceed basal relative inhomogeneity scales, whereas relative inhomogeneity values of planar-fit MinD and MinE density data, centered at pixel $(243, 124)$, during temporal partition P_9, significantly exceed basal relative inhomogeneity scales. The maximum relative inhomogeneity value of planar-fit MinD density data, centered at pixel $(436, 204)$, during the MinD density pulse upstroke in P_5, occurs at frame 21; the maximum relative inhomogeneity value of planar-fit MinD density data, centered at pixel $(243, 124)$, during temporal partition P_9, occurs at frame 52, when the MinD density traveling wave front passes through $D(243, 124, \bar{r})$. For comparison, planar-fit MinD density data, over $D(436, 204, \bar{r})$ at frame 21 of temporal partition P_5 and over $D(243, 124, \bar{r})$ at frame 52 of temporal partition P_9, are shown in Figure F.16.



(a) (b)

Figure F.16: Planar-fit MinD density data. Planar-fit MinD density data over $D(436, 204, \bar{r})$ at frame 21 of temporal partition P_5 is shown in (a), and planar-fit MinD density data over $D(243, 124, \bar{r})$ at frame 52 of temporal partition P_9 is shown in (b). Density data is overlaid on top of planar-fit density data with reduced opacity.

I consider planar-fit MinD and MinE density data over $D(436, 204, \bar{r})$ during P_5 to be near-homogeneous. Thus, I consider MinD and MinE density data over $D(436, 204, \bar{r})$ during P_5 to represent near-homogeneous processes with local pixel-to-pixel noise, and I approximate spatially homogeneous MinD and MinE density data by mean MinD and MinE density data over $D(436, 204, \bar{r})$ during P_5,

$$d_l = \mu^{\bar{r}} \circ d^{5,l}_{436,204}, \tag{F.28a}$$

$$e_l = \mu^{\bar{r}} \circ e^{5,l}_{436,204}, \tag{F.28b}$$

for $l \in \{1, 2, \ldots, n(P_5)\}$. Spatially near-homogeneous MinD and MinE density data profiles, d_l and e_l for $l \in \{1, 2, \ldots, n(P_5)\}$, are shown in Figure F.17.

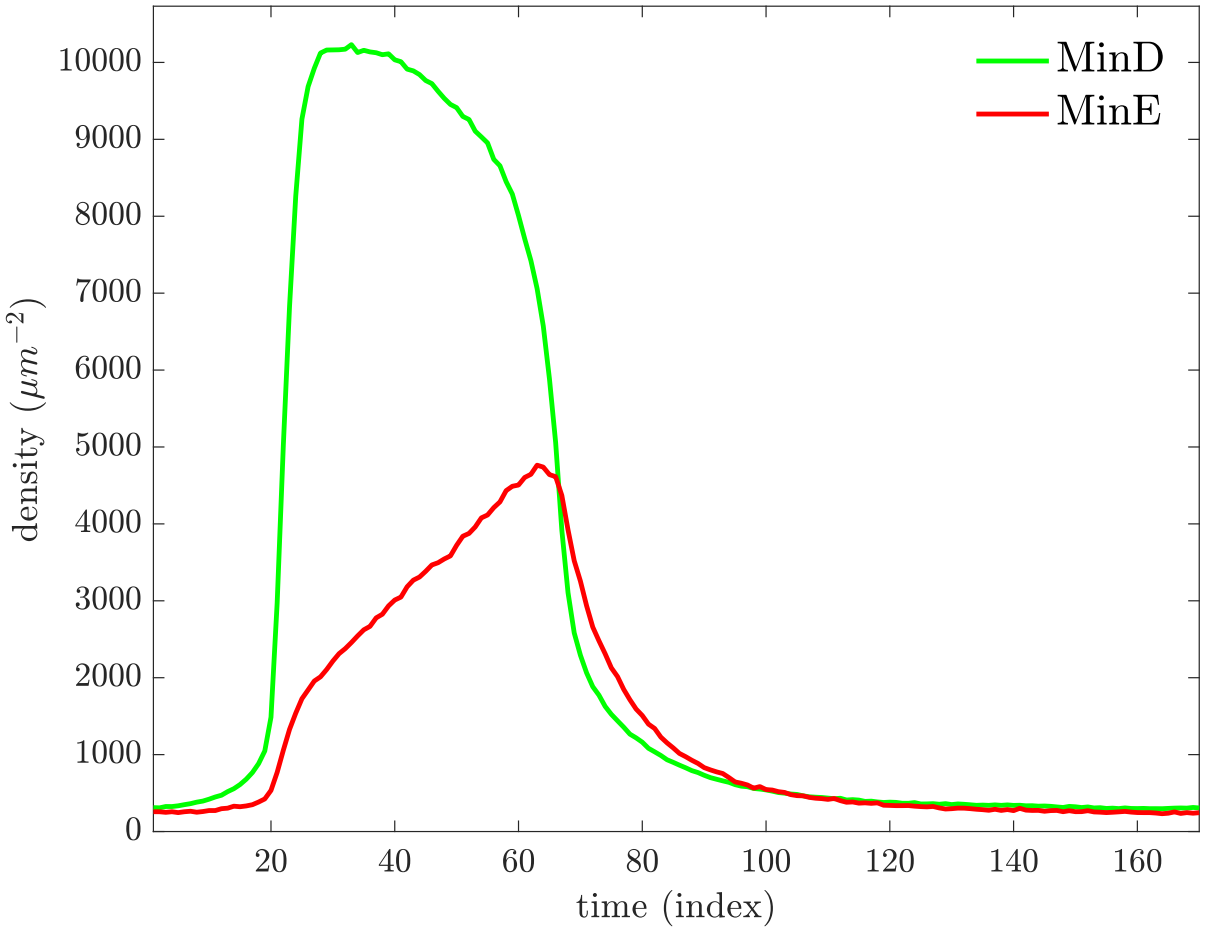


Figure F.17: Spatially near-homogeneous MinD and MinE density data profiles, d_l and e_l for $l \in \{1, 2, \ldots, n(P_5)\}$.

F.4.3 Errors in Spatially Near-Homogeneous Data

To show the spreads of MinD and MinE density data over $D(436, 204, \bar{r})$ during P_5, I plot normalized histograms of MinD and MinE density data over $D(436, 204, \bar{r})$ for each frame of P_5 in Figure F.18.

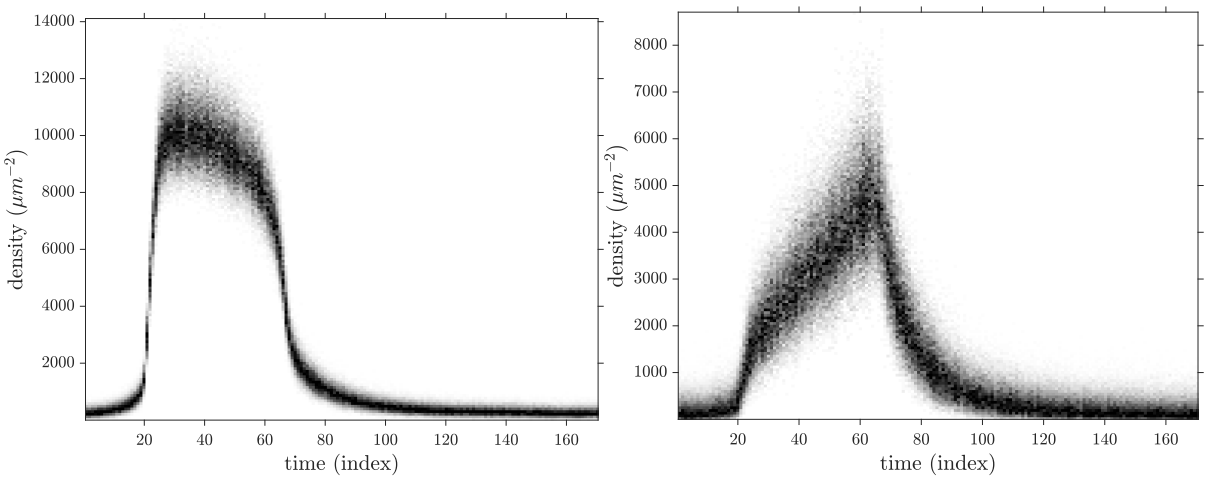


Figure F.18: The spreads of MinD and MinE density data over $D(436, 204, \bar{r})$ during P_5. Normalized histograms of density data over $D(436, 204, \bar{r})$ for each frame of P_5 are shown for MinD in (a) and MinE in (b). Each histogram is normalized by the maximum count in the histogram. Gradation is from white, with a count of 0, to black, with the maximum count in the histogram.

I approximate spatially near-homogeneous density data by mean density data. Thus, I calculate errors in approximating spatially near-homogeneous data by the standard error of the mean; for MinD and MinE density data over $D(436, 204, \bar{r})$ during P_5,

$$d_l^\sigma = (\pi \bar{r}^2)^{-\frac{1}{2}} \left(\sum_{m=-\bar{r}}^{\bar{r}} \sum_{n=-\bar{r}}^{\bar{r}} \frac{\rho_{436+m,204+n}}{\pi \bar{r}^2} \left(d^{5,l}_{436+m,204+n} - \mu^{\bar{r}} \circ d^{5,l}_{436,204} \right)^2 \right)^{\frac{1}{2}}, \tag{F.29a}$$

$$e_l^\sigma = (\pi \bar{r}^2)^{-\frac{1}{2}} \left(\sum_{m=-\bar{r}}^{\bar{r}} \sum_{n=-\bar{r}}^{\bar{r}} \frac{\rho_{436+m,204+n}}{\pi \bar{r}^2} \left(e^{5,l}_{436+m,204+n} - \mu^{\bar{r}} \circ e^{5,l}_{436,204} \right)^2 \right)^{\frac{1}{2}}, \tag{F.29b}$$

for $l \in \{1, 2, \ldots, n(P_5)\}$, where $\rho_{i+m,j+n}$ is the fraction of the $(i+m, j+n)^{\text{th}}$ pixel contained within $D(i, j, r)$. I show densities within $d_l \pm d_l^\sigma$ and $e_l \pm e_l^\sigma$ in Figure F.19.

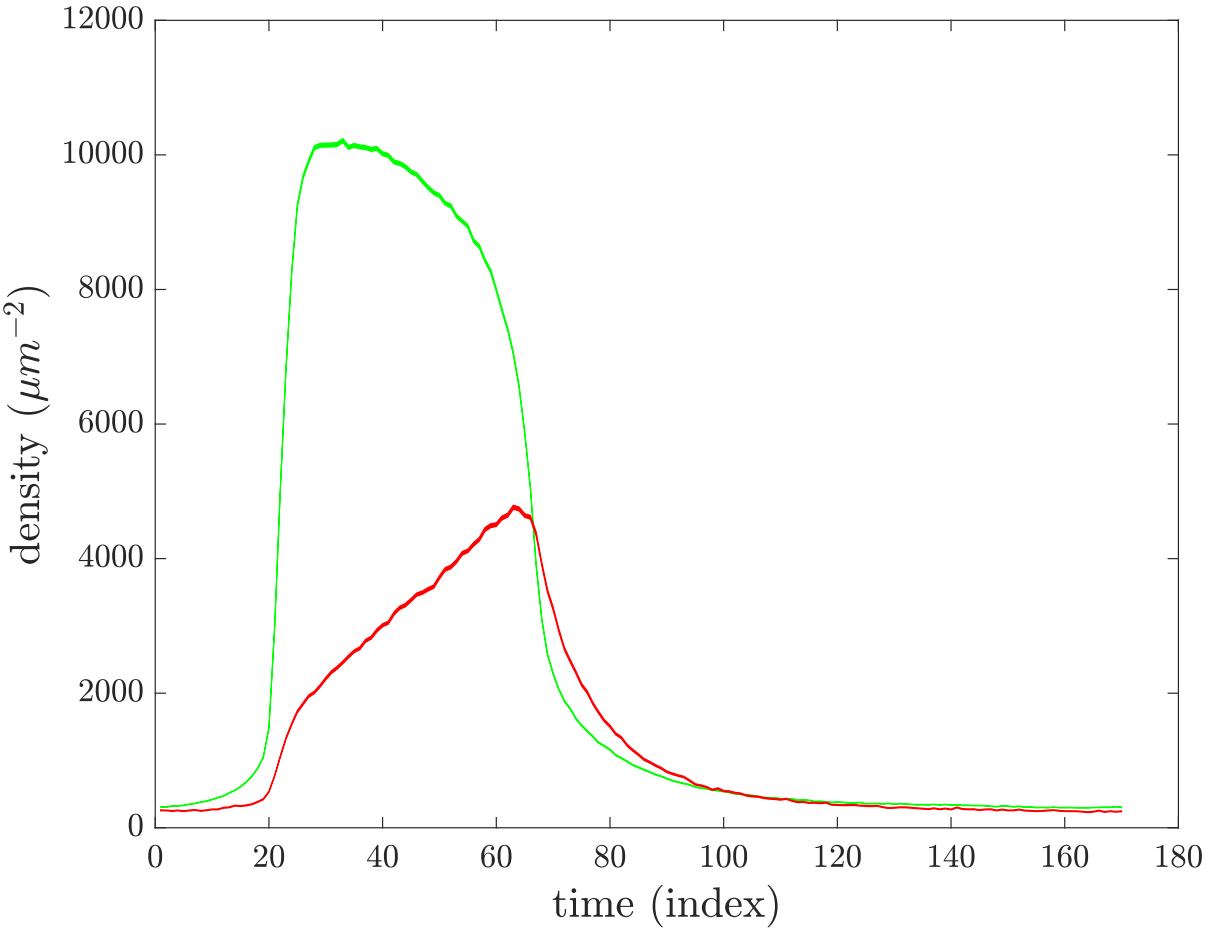
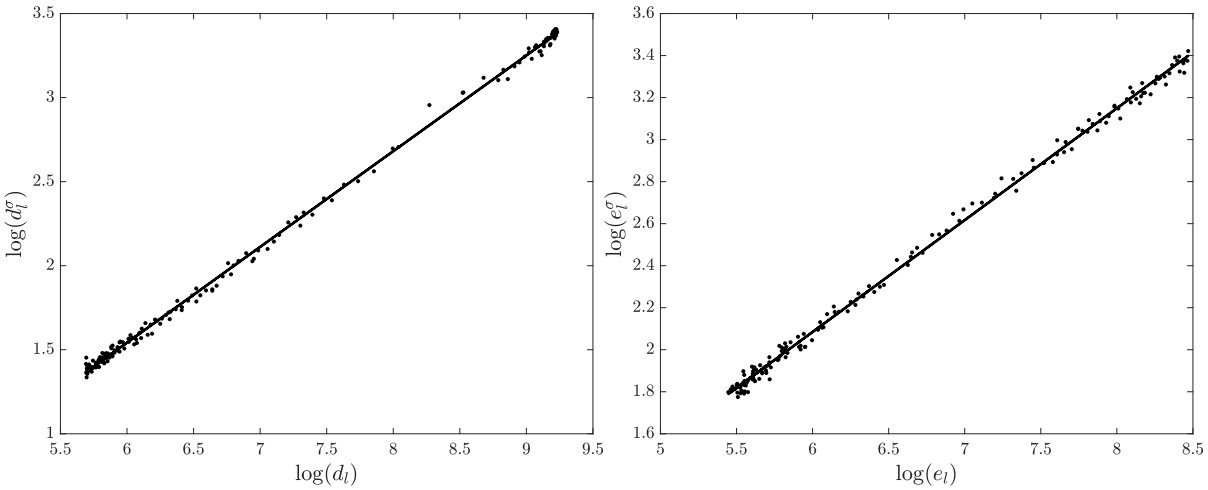


Figure F.19: Densities within error of spatially near-homogeneous data. Densities within $d_l \pm d_l^\sigma$ are shown in green, and densities within $e_l \pm e_l^\sigma$ are shown in red, for $l \in \{1, 2, \ldots, n(P_5)\}$

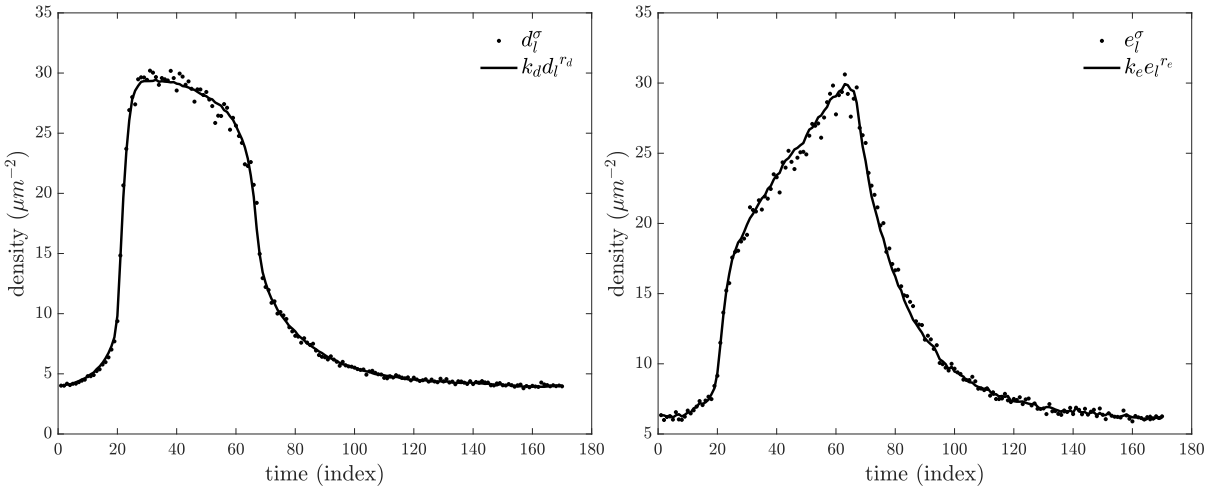
Interestingly, I find that d_l^σ and e_l^σ are related to d_l and e_l by power laws, $d_l^\sigma \approx k_d {d_l}^{r_d}$ and $e_l^\sigma \approx k_e {e_l}^{r_e}$ for constants k_d, r_d, k_e, and r_e. I fit $\log(k_d) + r_d \log(d_l)$ to $\log(d_l^\sigma)$ and $\log(k_e) + r_e \log(e_l)$ to $\log(e_l^\sigma)$ for $l \in \{1, 2, \ldots, n(P_5)\}$ using least squares, as shown in Figure F.20, to find $k_d = 0.153$, $r_d = 0.570$, $k_e = 0.330$, and $r_e = 0.532$.



(a) (b)

Figure F.20: Estimating power laws in errors. The least squares fit of $\log(k_d) + r_d \log(d_l)$ to $\log(d_l^\sigma)$ is shown in (a) and the least squares fit of $\log(k_e) + r_e \log(e_l)$ to $\log(e_l^\sigma)$ is shown in (b), for $l \in \{1, 2, \ldots, n(P_5)\}$. Log-errors are shown with points and fits are shown with lines.

I plot d_l^σ with $k_d {d_l}^{r_d}$ and e_l^σ with $k_e {e_l}^{r_e}$, with fit k_d, r_d, k_e, and r_e, in Figure F.21.



(a) (b)

Figure F.21: Spatially near-homogeneous data errors and power law approximations. d_l^σ and $k_d {d_l}^{r_d}$ are shown in (a), and e_l^σ and $k_e {e_l}^{r_e}$ are shown in (b), for $l \in \{1, 2, \ldots, n(P_5)\}$

F.4.4 Bounding Persistent and Bulk Densities

In MinD and MinE density data, higher densities than surrounding areas persist over time in some small regions, a phenomena likely related to the experimental observation that, in the

absence of MinE, a fraction of lipid bilayer-bound MinD resists membrane dissociation when washed with buffer, as discussed in the Ivanov and Mizuuchi Supporting Information. MinD and MinE densities consist of bulk densities, persistent lipid bilayer-bound densities, and transient lipid bilayer-bound densities. Thus, I include terms that account for bulk and persistent lipid bilayer-bound densities in mathematical models.

MinD and MinE pulse-train density data, generated by calculating mean values of MinD and MinE density data over $D(436, 204, \bar{r})$ during temporal partitions P_4, P_5, P_6, and P_7, is shown in Figure F.22.

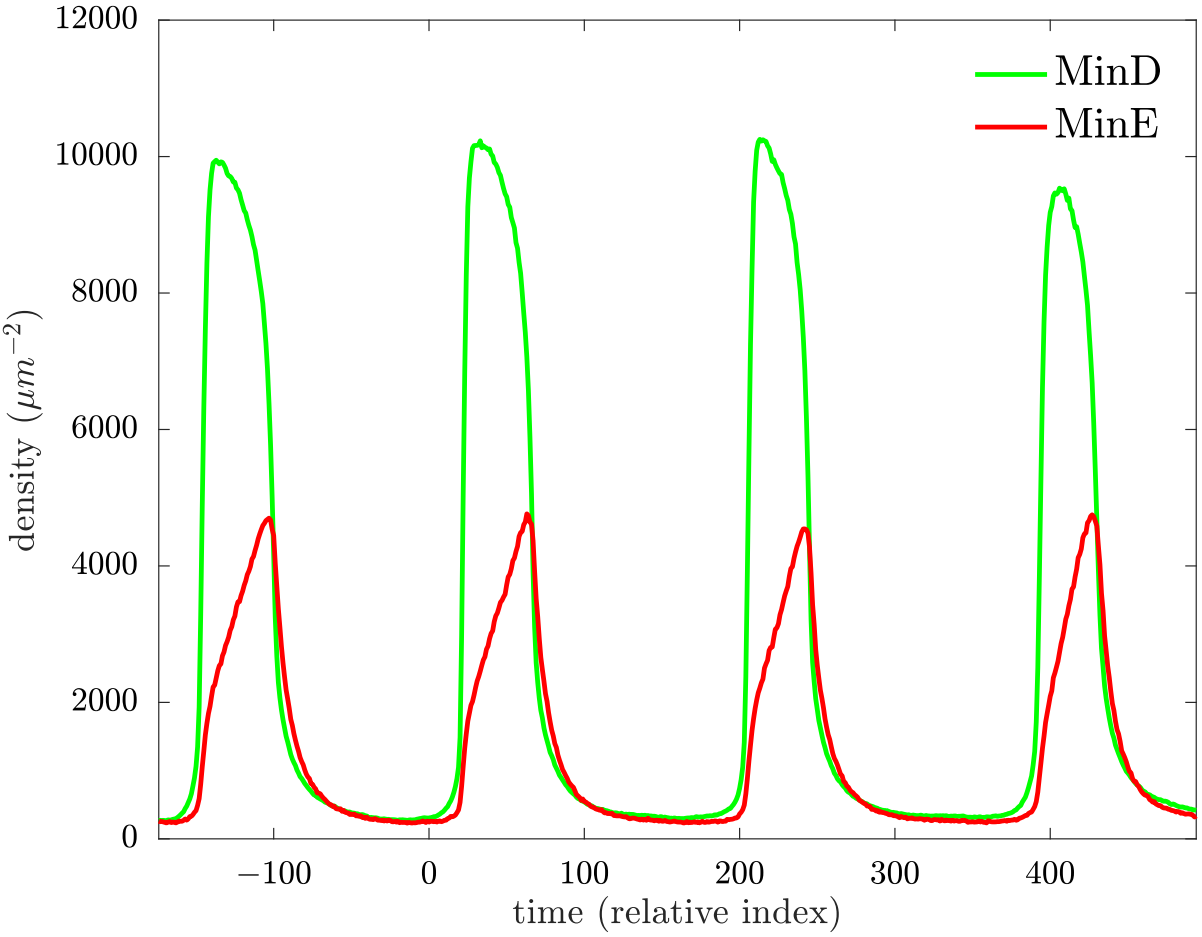


Figure F.22: MinD and MinE pulse-train density data. Temporal indexing corresponds to frame indexing in temporal partition P_5, as in Figure F.17.

As discussed previously, MinD and MinE pulse-train density generating data is near homogeneous during P_5. MinD and MinE pulse-train density generating data is near homogeneous during P_4, P_6, and P_7, except during MinD pulse upstrokes (with maximum planar-fit MinD relative inhomogeneity values of $2.17 \cdot \mu_d^h$, $3.10 \cdot \mu_d^h$, and $5.24 \cdot \mu_d^h$ and maximum planar-fit MinE relative inhomogeneity values of $1.10 \cdot \mu_e^h$, $1.20 \cdot \mu_e^h$, and $2.28 \cdot \mu_e^h$). As is visible in Figure F.22, consecutive MinD and MinE density pulses are similar in dynamic behavior, apart from relatively small

differences in peak density values and pulse periods, which may result from spatial asymmetries during pulse upstrokes. Repeatedly in the pulse train, MinD and MinE densities sharply increase during pulse upstrokes, peak, sharply decrease, slowly decay, level off to minimal values, then slightly increase before pulse upstrokes. I find minimal MinD and MinE density data between successive MinD pulse-train density peaks, the MinD and MinE density data with least means over 17 consecutive data indices, 1/10 the number of data indices in P_5. Between the first and second, second and third, and third and fourth MinD pulse-train density peaks, minimal MinD and MinE densities are roughly constant, with minimal MinD density means of 277.16 μm^{-2}, 302.79 μm^{-2} and 317.88 μm^{-2}, minimal MinD density standard deviations of 3.18 μm^{-2}, 4.59 μm^{-2}, and 4.18 μm^{-2}, minimal MinE density means of 240.68 μm^{-2}, 243.20 μm^{-2}, and 249.25 μm^{-2}, and minimal MinE density standard deviations of 6.58 μm^{-2}, 6.99 μm^{-2}, and 7.68 μm^{-2}. Mean minimal MinD and MinE densities increase during the pulse train, but by relatively insignificant amounts on the scales of MinD and MinE density pulses.

Bulk MinD and MinE densities are roughly constant during the pulse-train. Given the similarities in dynamic behavior and minimal values of successive MinD and MinE density pulses, persistent and transient lipid bilayer-bound MinD and MinE densities likely attain similar minimal values during successive pulses of the pulse train. Thus, as persistent lipid bilayer-bound densities are inherently temporally non-decreasing, persistent lipid bilayer-bound MinD and MinE densities are likely roughly constant, on the scales of MinD and MinE density pulses, during temporal partition P_5. As such, for spatially near-homogeneous MinD and MinE density data, I model the sums of bulk and persistent lipid bilayer-bound MinD and MinE densities as constants, C_d and C_e. MinD and MinE densities consist of bulk densities, persistent lipid bilayer-bound densities, and transient lipid bilayer-bound densities. Thus, minimal MinD and MinE densities are upper bounds of persistent lipid bilayer-bound MinD and MinE densities. As such, I impose upper bounds on C_d and C_e, as the maximum values of minimal MinD and MinE density means, 317.88 μm^{-2} and 249.25 μm^{-2}.

Appendix G

Implementation of Overlapping-Niche Descent for Near-Homogeneous Data Fitting

Here, I describe details pertaining to the implementation of overlapping-niche descent for model fitting to the near-homogeneous data. I describe related structural components of overlapping-niche descent in Section 4.4.

G.1 Generating Random Parameter and State Values

Initially in overlapping-niche descent, I randomly generate parameters and state values. Also, as discussed in Section C.1, throughout overlapping-niche descent, I randomly generate parameters and state values in random offspring. In accordance with bounds on C_d, C_e, and $c_{\bar{d}}$ (4.16) and (4.17), I randomly generate values of C_d, C_e, and $c_{\bar{d}}$ such that

$$C_d \sim 317.88 \cdot U(0,1)\ \mu\text{m}^{-2}, \tag{G.1a}$$

$$C_e \sim 249.25 \cdot U(0,1)\ \mu\text{m}^{-2}, \tag{G.1b}$$

$$c_{\bar{d}} \sim 158.94 \cdot U(0,1)\ \mu\text{m}^{-2}, \tag{G.1c}$$

where $U(a,b)$ is the uniform probability distribution over the interval (a,b). I expect that $c_{\max}$ is within one or two orders of magnitude of half the maximal near homogeneous MinD density value, $D_{\max}/2$. Thus, in accordance with bounds on $c_{\max}$ (4.18), I randomly generate $c_{\max}$ such that

$$c_{\max} \sim D_{\max}/2 \cdot 10^{U(0,2)}. \tag{G.2}$$

Additionally, I expect that c_s is within one or two orders of magnitude of $D_{\max}/2$. Thus, in accordance with bounds on c_s (4.19), I randomly generate c_s such that

$$c_s \sim D_{\max}/2 \cdot 10^{U(-2,0)}. \tag{G.3}$$

Given no prior parameter value estimates, I randomly generate rate parameters over a broad range of scales:

$$p \sim 10^{U(-9,1)}\ u_p \text{ for all } p \in \left\{\omega^z_{u,v\to x,y} : u,v,x,y,z \in \{\emptyset, D, E, d, de, ede, ded, e\}\right\}, \tag{G.4}$$

where u_p is the units of parameter p.

I choose random state values to match near homogeneous data exactly. For near-homogeneous MinD and MinE data values, D and E, $\bar{D} = (D - C_d)/2$, and $\bar{E} = (E - C_e)/2$, with c_e as a free state, I generate random state values for the modified Bonny *et al* model and the extended Bonny *et al* model such that

$$\begin{aligned} c_e &\sim U(\max\{0, \bar{E} - \bar{D}\}, \bar{E}), \\ c_{de} &= \bar{E} - c_e, \\ c_d &= \bar{D} - \bar{E} + c_e. \end{aligned} \tag{G.5}$$

With c_{ede} and c_e as free states, I generate random state values for the symmetric activation model such that

$$\begin{aligned} c_{ede}, c_e &\sim U\left(\left\{c_{ede} + c_e \geq \bar{E} - \bar{D}, 2c_{ede} + c_e \leq \bar{E} : c_{ede} \geq 0, c_e \geq 0\right\}\right), \\ c_{de} &= \bar{E} - 2c_{ede} - c_e, \\ c_d &= \bar{D} - \bar{E} + c_{ede} + c_e, \end{aligned} \tag{G.6}$$

where $U(\{\cdot\})$ is the uniform probability distribution over the set $\{\cdot\}$. With c_{ded} and c_e as free states, I generate random state values for the asymmetric activation model such that

$$\begin{aligned} c_{ded}, c_e &\sim U\left(\left\{c_e - c_{ede} \geq \bar{E} - \bar{D}, c_{ede} + c_e \leq \bar{E} : c_{ded} \geq 0, c_e \geq 0\right\}\right), \\ c_{de} &= \bar{E} - c_{ded} - c_e, \\ c_d &= \bar{D} - \bar{E} - c_{ded} + c_e, \end{aligned} \tag{G.7}$$

G.2 Parents and Offspring

I choose parents and offspring as in Section E.1.2. For convenience, I repeat the discussion from Section E.1.2 below. The function of parents and offspring in overlapping-niche descent is described in Section C.1. Accordingly, to the i^{th} niche in generation g, I allocate one sustained parent, $\hat{n}_i = 1$, one high momentum offspring, $\check{n}^m_{g,i} = 1$, one cross-niche offspring, $\check{n}^c_{g,i} = 1$, and one random offspring, $\check{n}^r_{g,i} = 1$, for each $i \in \{1, 2, \ldots, 101\}$ and each generation of overlapping-niche descent, $g \geq 1$. In the first two generations of overlapping-niche descent, $g \leq 2$, I allocate two sexual offspring to each niche, $\check{n}^s_{g,i} = 2$ for all $i \in \{1, 2, \ldots, 101\}$. After the second generation of overlapping-niche descent, I adaptively change the number of sexual offspring that I allocate to each niche, enlarging less convergent niches and shrinking more convergent

niches for greater efficiency in optimization. Specifically, I allocate one sexual offspring to the i^{th} niche, and randomly allocate the remaining 101 sexual offspring to the i^{th} niche with probability proportional to $\Delta r_{g,i,1}$, the measure of convergence in the (first) parent space of the i^{th} niche in generation g, as defined in equation (C.1), for each $i \in \{1, 2, \ldots, 101\}$ and $g > 2$.

G.3 Selection and Random Perturbation

I choose the natural default value for the selection strength parameter, $q_{\text{fit}} = 1$, for q_{fit} as described in Section C.1. For a sexual offspring that inherits parameter p from individual $(\mathbf{p}_{g,i,j}, \mathbf{x}_{g,i,j})$, I perturb the value of the parental parameter, $\hat{p}$, such that

$$p \sim \begin{cases} \left(\hat{p} + \hat{p} \cdot N\left(0, \max\{\Delta r_{g,i,j}, 10^{-2}\}^2\right) \;\middle|\; p_{\min} \leq p \leq p_{\max}\right) & \text{if } p \in \{C_d, C_e, c_{\bar{d}}\} \\ \left(\hat{p} \cdot 10^{N\left(0, \max\{\Delta r_{g,i,j}, 10^{-2}\}^2\right)} \mid p_{\min} \leq p \leq p_{\max}\right) & \text{otherwise,} \end{cases} \tag{G.8}$$

where $N(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2, $\Delta r_{g,i,j}$ is the measure of convergence in the j^{th} parent space of the i^{th} niche in generation g (C.1), $p_{\min}$ is the restricted lower bound on parameter p as discussed in Section 4.4.3, and $p_{\max}$ is the restricted upper bound on parameter p as discussed in Section 4.4.3. A standard deviation of $\max\{\Delta r_{g,i,j}, 10^{-2}\}$ ensures some small but significant perturbation in parameter p when $\Delta r_{g,i,j}$ is small. Similarly, for a sexual offspring that inherits state value x from individual $(\mathbf{p}_{g,i,j}, \mathbf{x}_{g,i,j})$, I perturb the value of the parental state value, $\hat{x}$, such that

$$x \sim \left(\hat{x} + \hat{x} \cdot N\left(0, \max\{\Delta r_{g,i,j}, 10^{-2}\}^2\right) \;\middle|\; x \geq 0\right). \tag{G.9}$$

Details pertaining to sexual offspring are described in Section C.1.

G.4 Dykstra's Method

For fits to near-homogeneous data, restrictions on parameters and state values, inequalities (4.16), (4.17), (4.18), (4.19), (4.20), (4.21), (4.22), and (4.23), can be written as a collection of linear inequalities. Thus, during accelerated descent, I employ projection using Dykstra's method, as discussed in Section C.2.3. For Dykstra's method, I choose a small relative termination tolerance, $\varepsilon_c = 10^{-6}$, and a smaller absolute termination tolerance, $\varepsilon_{\bar{c}} = 10^{-12}$. To avoid overly slowing accelerated descent from a large number of projections, I prematurely terminate accelerated descent if the number of iterations in Dykstra's method exceeds 10^4.

G.5 Initial values, Termination, Prolongation, and Computation

I choose values as in Section E.1.5. For convenience, I repeat the discussion from Section E.1.5 below (details relating to computation differ from those in Section E.1.5). I choose the initial gradient scaling value $s_{i,0} = 0$, for all i in the indexed set of all parameters and state values. Details pertaining to $s_{i,0}$ are described in Section C.2.1. I choose the maximum number of strict descent iterations to be relatively but not excessively large, $n_{\max} = 10^4$, to ensure sufficient convergence to a local minimum of $r(\mathbf{p}, \mathbf{x}; \lambda)$ while avoiding overburdensome computation. I choose a very small contraction termination tolerance, $\varepsilon_\sigma = 10^{-30}$, and a very small relative-change termination tolerance, $\varepsilon_r = 10^{-30}$, to continue accelerated descent through $n_{\max}$ strict descent iteration unless local minimization is essentially complete. Details pertaining to $n_{\max}$, ε_σ, and ε_r are described in Section C.2.2. For descent prolongation, I choose: $\breve{\sigma} = 1$, for non-stringent descent prolongation, $m_{\text{pro}} = 10^3$, a factor of 10 less than $n_{\max}$; $\hat{n}_{\text{pro}} = n_{\max} = 10^4$; and $\check{n}_{\text{pro}} = n_{\max} = 10^4$. Details pertaining to $\breve{\sigma}$, m_{pro}, $\hat{n}_{\text{pro}}$, and $\check{n}_{\text{pro}}$ are described in Section C.3. I choose the overlapping-niche descent termination tolerance to be relatively but not exceedingly small, $\varepsilon_{\Delta r} = 10^{-3}$, for reliable convergence in all niches while avoiding an excessive number of overlapping-niche descent generations. Details pertaining to $\varepsilon_{\Delta r}$ are described in Section C.1. I compute genetic algorithm calculations using *MATLAB*. I compute accelerated descent calculations in parallel using *C++* on the Calcul Québec server Guillimin, the WestGrid server Orcinus, the Compute Canada server Cedar, and the Compute Canada server Graham.