Parameter Estimation for Systems of Ordinary Differential Equations

by

Jonathan Calver

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

Abstract

Parameter Estimation for Systems of
Ordinary Differential Equations

Jonathan Calver
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
2019

We consider the formulation and solution of the inverse problem that arises when fitting systems of ordinary differential equations (ODEs) to observed data. This parameter estimation task can be computationally intensive if the ODEs are complex and require the use of numerical methods to approximate their solution. The thesis focuses on addressing a common criticism of the single shooting approach, which is that it can have trouble converging to the globally optimal parameters if a sufficiently good initial guess for the parameters is unavailable. We demonstrate that it is often the case that a good initial guess can be obtained by solving a related inverse problem. We show that gradient based shooting approaches can be effective by investigating how they perform on some challenging test problems from the literature. We also discuss how the approach can be applied to systems of delay differential equations (DDEs). We make use of parallelism and the structure of the underlying ODE models to further improve the computational efficiency.

Some existing methods for the efficient computation of the model sensitivities required by a Levenberg-Marquardt least squares optimizer are compared and implemented in a parallel computing environment. The effectiveness of using the adjoint approach for the efficient computation of the gradient required by a BFGS optimizer is also investigated for both systems of ODEs and systems of DDEs. The case of unobserved components of the state vector is then considered and we demonstrate how the structure of the model and the observed data can sometimes be exploited to cope with this situation.

In memory of my grandfather, Ozzie Weninger.

Acknowledgements

I feel very fortunate to have had the opportunity and privilege to complete my graduate studies in the NA group of the CS department at UofT under the supervision of Wayne Enright. I could not have asked for a better supervisor.

I would like to thank Christina Christara and Ken Jackson for being on my thesis committee, as well as Maryam Mehri Dehnavi and Tom Fairgrieve for joining my final oral committee. The time you have taken to provide me with feedback on this thesis is greatly appreciated. I would also like to thank Allan Willms for accepting to be the external examiner. Your insightful appraisal and detailed comments helped to improve the final presentation of this thesis.

Special thanks to the other grad students in the NA group and in the department, with whom I have had the pleasure to spend this time. Lastly, I would like to thank my family. Your support and encouragement have meant the world to me.

Contents

List of Tables

List of Figures

Abbreviations

ACCEL accelerated least squares. 17

AINT-SME algrebraic integral smooth and match estimator. 84

ASME algrebraic smooth and match estimator. 83

CD centered difference. 47

CE Cross Entropy. 42

CRK continuous Runge-Kutta. 5

DBE numerical discretization-based estimation. 11

FD forward difference. 47

GFM Green's function method. 53

GN Gauss-Newton. 44

INT-SME integral smooth and match estimator. 13

IVP initial value problem. 2

LHS latin hypercube sampling. 9

LM Levenberg-Marquardt. 43

LPE local polynomial estimation. 11

LS linear least squares. 13

MLE maximum likelihood estimation. 3

NLS non-linear least squares. 3

NM Newton's method. 45

PMM piecewise Magnus method. 54

RK Runge-Kutta. 5

SME smooth and match estimator. 10

SSE sum of squared errors. 1

WLS weighted least squares. 12

Chapter 1

Introduction

1.1 Motivation

Throughout the sciences, mathematical models are developed to help us better understand the world around us. In order to accurately reproduce complex phenomena, such models are often nonlinear and do not permit closed form analytical solutions. This means that the models must be approximately solved using numerical methods implemented on computers. In this thesis, we consider models formulated as parameterized initial value problems (IVPs). These models consist of model dynamics specified by a system of ordinary differential equations (ODEs) and initial conditions specifying the initial state of the model. IVPs are used in a wide range of applications, including Lotka-Volterra predator-prey models of population dynamics [6], enzyme kinetics [19, 18], the spread of disease [26], chemical reactions [30], neuron signalling [16, 53], and many more. The solution of an IVP is approximated by simulating the model state from the initial time to some final time of interest. The solution of the IVP over an interval of interest is referred to as a trajectory. Numerous numerical methods exist for approximating these trajectories. In this thesis, we focus on the use of high order reliable Continuous Runge-Kutta (CRK) methods, which are appropriate for ODEs that are at most mildly stiff.

Due to model simplifications and measurement noise, models tend not to exactly fit observed data. Parameter estimation seeks to find the set of model parameters such that the model best fits the collected data, as defined by an appropriate objective function. For example, the sum of squared errors (SSE) between the data and the model prediction is often used. Parameter estimation is often referred to as an inverse problem.

Estimating the best fit parameters can be computationally intensive. In the case of IVPs, evaluating the SSE for a candidate set of model parameters requires a trajectory

simulation. If the model is complex or the interval of interest is large, this simulation can be computationally expensive. This observation has led to a variety of techniques being developed to reduce the number of model trajectory simulations required to estimate the best fit model parameters. In this thesis, we describe several of these techniques and demonstrate how they can be used to reduce the computational cost of performing parameter estimation for IVPs.

1.2 Definitions and Notation

We now introduce notation used throughout the thesis. Additional notation will be introduced in specific sections as it is needed.

1.2.1 ODE definition

We consider the parameterized initial value problem (IVP),

$$\begin{aligned} y'(t) &= f(t, y(t), p), \\ y(0) &= y_0, \\ t &\in (0, T), \end{aligned} \tag{1.1}$$

where $y(t)$ is the state vector of dimension n_y, p is a constant vector of model parameters of dimension n_p, and y_0 are the initial conditions of the state vector, $y(0)$. For ease of notation, we assume each component of y_0 is a model parameter that appears in p. We will denote the solution of (1.1) for a specific p by $y(t, p)$.

In general, f is a nonlinear map from $\mathcal{R}^1 \times \mathcal{R}^{n_y} \times \mathcal{R}^{n_p} \to \mathcal{R}^{n_y}$. In some applications, a subset of the parameters only appear linearly in f and one can often exploit this structure. In these situations,

$$p = [q,\ r,\ y_0], \tag{1.2}$$

where q is the vector of n_q parameters that appear nonlinearly in f and r is the vector of n_r parameters that only appear linearly in f (e.g. rate constants in chemical kinetics). This allows us to decompose $f(t, y(t), p)$ as,

$$f(t, y(t), p) = G(t, y(t), q)r + g(t, y(t), q), \tag{1.3}$$

where G maps $\mathcal{R} \times \mathcal{R}^{n_y} \times \mathcal{R}^{n_q} \to \mathcal{R}^{n_y} \times \mathcal{R}^{n_r}$ and g maps $\mathcal{R} \times \mathcal{R}^{n_y} \times \mathcal{R}^{n_q} \to \mathcal{R}^{n_y}$.

1.2.2 DDE definition

We also consider IVPs where the underlying system of ODEs is replaced with a system of delay differential equations (DDEs). In this thesis, we restrict ourselves to systems of DDEs with constant lags, in which case we are considering the IVP,

$$\begin{aligned} y'(t) &= f(t, y(t), y(t-\tau_1), \ldots, y(t-\tau_{n_d}), p), \\ y(t) &= h(t,p)\ ;\ t < 0, \\ t &\in (0, T), \end{aligned} \tag{1.4}$$

where each τ_r, $r = 1, \ldots, n_d$, is a constant delay that appears in the parameter vector, p, and $h(t,p)$ is a history function. Note that, as with (1.1), y and h are vector valued functions of dimension n_y.

1.2.3 Least Squares Parameter Estimation

We assume that a set of observations of the state vector is known and given by,

$$\hat{y}_j(t_i) = y_j(t_i) + \mathcal{N}(0, \sigma_{ij}^2), \text{ for } \ i = 1, \ldots, n_o;\ j = 1, \ldots, n_y, \tag{1.5}$$

where n_o is the number of observation points, $y_j(t_i)$ denotes the j^{th} component of the true state vector at time t_i, and $\mathcal{N}(0, \sigma_{ij}^2)$ is normally distributed noise with variance σ_{ij}^2. In some applications, the state vector is not directly observed, but rather the observations are functions of the state vector, specified by a measurement function. We do not consider such cases in this thesis.

Given such data, parameter estimation for ODEs is traditionally done using maximum likelihood estimation (MLE). This leads to solving the following non-linear least squares (NLS) problem,

$$\min_p \sum_{i=1}^{n_o} \sum_{j=1}^{n_y} \frac{(\hat{y}_j(t_i) - y_j(t_i, p))^2}{2\sigma_{ij}^2}, \text{ such that } y(t,p) \text{ satisfies (1.1).} \tag{1.6}$$

If the σ_{ij}'s are not known, an iterative reweighted approach may be used to solve this problem. Throughout this thesis, we will make the simplifying assumption that all observations have the same variance, so each $\sigma_{ij} = \sigma$, where σ is a constant.

Given this simplification, the objective function (1.6) reduces to,

$$O(p) = \sum_{i=1}^{n_o} \sum_{j=1}^{n_y} \frac{(\hat{y}_j(t_i) - y_j(t_i, p))^2}{2}, \tag{1.7}$$

and the parameter estimation problem we consider is to determine the optimal value of p, call it $\hat{p}$, such that $y(t,p)$ satisfies (1.1) and minimizes $O(p)$. We focus on models and data for which this problem is well posed. The issues of structural and practical identifiability [41] are important themselves, but they are not directly investigated in this thesis. There are two ways to view this NLS optimization problem (1.7), which have received considerable attention in the literature. We will primarily consider the single shooting approach [4], where an IVP solver is used to approximate the trajectory of $y(t,p)$, to within a user specified tolerance, whenever we evaluate the objective function. That is, we satisfy (1.1) on each iteration of the optimization. Alternatively, approaches like Ramsay's functional data analysis [40] can be used. In these approaches, the ODE constraint is not strictly satisfied on each iteration during the optimization, but it is instead treated by the inclusion of a penalty term in the objective function, which will ensure that, at convergence, $y(t,p)$ satisfies the IVP. This approach is discussed in Section 2.5.3.

In order to solve (1.7) using a gradient based optimizer and the single shooting approach, we require the sensitivity information,

$$\frac{\partial O}{\partial p} = J(p) = \sum_{i=1}^{n_o}\sum_{j=1}^{n_y} -\frac{\partial y_j}{\partial p}(t_i)\big(\hat{y}_j(t_i) - y_j(t_i,p)\big). \tag{1.8}$$

As we will discuss in detail in Chapter 3, this requires us to either approximate the model sensitivities, $\frac{\partial y}{\partial p}$, at each observation point, t_i, or approximate $J(p)$ directly using an approach such as the adjoint method, which we discuss in Chapter 4.

A common criticism of using a gradient based optimizer and the single shooting approach is that, since the problem is nonlinear, it relies on the user being able to specify an initial p, call it p_0, that is sufficiently close to the best fit $\hat{p}$. In Chapter 2, we discuss several ways to use the observed data and structure of the model to obtain a suitable p_0.

1.2.4 Approximating Model Sensitivities

There are several ways to accurately approximate the required model sensitivities. In this thesis, we consider the use of finite differences and two ways to approximate the variational equations. The variational equations can be directly approximated using a forward simulation of the variational IVP [15, 57] or a Green's Function method can be used [25, 28]. Depending on the level of accuracy required and the size of the problem, the relative performance of these methods will vary. Also, if parallelism is being employed, the relative costs of these methods will change. Other approaches that we do not consider

include the complex step method [37, 3], automatic differentiation [7], and the internal numerical differentiation (IND) method of Bock [31].

1.2.5 Adjoint Method

The adjoint method is distinguished in that it calculates $J(p)$ without explicitly calculating the model sensitivities. This is accomplished by introducing a vector of adjoint variables, $\lambda^T(t)$, which has the same dimension as $y(t)$. Since the adjoint method only calculates the vector $J(p)$ and not the matrix of model sensitivities, it cannot be used in conjunction with a Gauss-Newton type least squares optimization algorithm. Instead, it can be applied to directly determine the solution of $J(p) = 0$ using a quasi-Newton method, such as BFGS.

1.2.6 Numerical Solutions of IVPs

It is important to understand that the objective of a numerical algorithm for solving this parameter estimation problem is, when given a user specified accuracy parameter (TOL), to determine an approximate best fit solution, $\hat{p}_{approx}$, such that,

$$\|\hat{p} - \hat{p}_{approx}\| < K \text{ TOL}, \tag{1.9}$$

where K depends on the conditioning of the problem. In order to achieve this, we have to use a reliable ODE solver with a well justified step size control strategy, as well as a carefully chosen stopping criterion for the least squares optimization.

Many reliable methods exist for solving ODEs, but here we will briefly mention a continuous Runge-Kutta (CRK) solver that is particularly well suited to our application. This solver is a standard Runge-Kutta (RK) solver , except that it additionally computes a local interpolant between time steps. This allows for reliable high accuracy off-mesh interpolation of $y(t)$, as well as its derivative. In addition, for values of t between time steps, we define the defect of the local interpolant in terms of how well it satisfies the underlying ODE. The user specified tolerance, TOL, controls the defect of the computed solution, such that the method attempts to ensure it is bounded by a small multiple of TOL on each step. Throughout this thesis, we use the DDEM [55] package to simulate IVPs using a 6^{th} order CRK with defect control. In some situations we also use the standard ode45, ode15s, and dde23 solvers of MATLAB.

1.3 Contributions of the Thesis

The main contribution of the thesis is the development of a two stage algorithm for efficiently and robustly estimating the parameters in systems of ODEs that directly attempts to produce an approximate $\hat{p}_{\text{approx}}$ that satisfies (1.9). Our algorithm draws on ideas developed and rediscovered over the last 50 or more years. The algorithm we propose and justify is the only method for parameter estimation that we are aware of that directly and adaptively attempts to ensure that the approximation solution satisfies (1.9). We demonstrate the effectiveness of our approach on several test problems from the literature and emphasize how each stage of our two stage algorithm can be computed efficiently.

The first stage involves computing the initial guess, p_0, of the parameters. The proposed procedure is based on an approach dating back to Varah [49], which has been recently rediscovered and further developed in several disciplines. We apply the approach used by Dattner [14] to the case of nonlinear parameters and problems where the underlying IVP is a delay differential equation (DDE). We also emphasize how the structure of the system of ODEs can lead to even more computational efficiency.

The second stage involves finding the minimizer of (1.7), with p_0 coming from the first stage. To improve the computational efficiency of this stage, we investigate several methods for approximating $\frac{\partial y}{\partial p}(t)$. Through numerical experiments, we find that using the Levenberg-Marquardt algorithm is more efficient than either using BFGS and adjoint gradients or using a full Newton method. We then explore several approaches for approximating the Jacobian required by Levenberg-Marquardt and demonstrate how each approach can be efficiently computed in parallel. In particular, we find that the parallel Green's Function method can be more efficient than the parallel variational approach in some situations. Efficient computation of the gradient via the adjoint method is also investigated. We identify and propose ways to address a limitation of the adjoint method which arises when the number of observations is large.

We go on to discuss what can be done when data are not available for all components of the state vector. We consider several techniques for handling this difficulty and suggest an improvement to a method recently proposed by Dattner [13]. Other general ways to improve the efficiency of the parameter estimation task are also identified and their performance evaluated through numerical experiments.

1.4 Outline of the Thesis

In Chapter 2, we introduce our two stage estimation procedure and describe the first stage of our estimation procedure in detail. We present numerical results demonstrating the performance of the initial procedure when the quantity and quality of the observed data is varied. We then present numerical results demonstrating the performance of our two stage estimation procedure on several challenging test problems, including a DDE model. We also consider other ways to reduce the total computational cost. In Chapter 3, we discuss how to efficiently perform the minimization of (1.7) in the second stage of our estimation procedure. Several methods to compute the required model sensitivities are investigated and implemented in a parallel computing environment. In Chapter 4, we discuss the adjoint method for computing gradients and numerically explore its limitations when applied to ODEs and constant lag DDEs. Chapter 5 considers other techniques that can be used when data are not available for all components of the state vector. Chapter 6 concludes the thesis and discusses potential directions for future work.

Chapter 2

Two Stage Procedure

2.1 Motivation

A straightforward way to efficiently determine the minimizer of (1.7) is to use a gradient based optimizer. For example, since this is an NLS problem, one might use a Gauss-Newton or Levenberg-Marquardt [32] algorithm. In order to apply these methods, the user must provide an initial guess, p_0 for the vector of unknown parameters. If p_0 is not sufficiently close to $\hat{p}$, the optimizer may converge to a suboptimal local minimizer of the objective function or the optimizer may converge slowly. Furthermore, if very little is known about the parameters, it might be challenging just to find a p_0 such that the underlying ODE solver is able to successfully simulate the model over the entire interval of interest (and evaluate $O(p_0)$).

If a suitable p_0 is readily available, then determining the minimizer of (1.7) with a gradient based optimizer works quite well. In the case where it is somewhat difficult to find a suitable p_0, applying a multi-start method may be possible. Multi-start methods simply sample the parameter space and run a gradient based optimizer from each sample - starting with the most promising samples. When a suitable p_0 is not known, it is quite common to attempt to apply a heuristic based, derivative free, global search algorithm to minimize (1.7). While genetic algorithms, particles swarms, and other such algorithms are likely to eventually find a solution, they are often impractically slow. This has led to the development of hybrid optimizers, which combine a global search heuristic with a local optimizer. Hybrid optimizers run the global search algorithm and periodically attempt to run a local optimizer from a candidate p_0. If a sufficiently good minimizer of (1.7) is not found, the hybrid optimizer returns to the global search algorithm. Hybrid optimizers have been shown to speed up global search algorithms by several orders of magnitude [44] in some cases.

However, if the search space is large, these hybrid optimizers may still be computationally expensive. The high cost comes from two factors. First, global search algorithms repeatedly evaluate the objective function. In this application, evaluating the objective function requires reliably simulating a system of ODEs. Second, as the number of parameters increases, the size of the search space grows exponentially. Since global search algorithms generally treat the objective function as a black box, a natural question is whether or not the structure of the objective function can be exploited to aid in finding the minimizer of (1.7). Specifically, we will look at how the structure of the ODE, along with the observed data, can allow alternative, less expensive minimization problems to be formulated, whose minimizers are good initial guesses for the minimizer of (1.7).

2.1.1 Log Hypercube Sampling

Latin hypercube sampling (LHS) [42] is a commonly used technique for sampling. For our application, it has been suggested that it is better to sample from the log of the parameter space [36] (that is, if the parameter space is $[10^{-b}, 10^{b}]^{n_p}$, then we perform Latin hypercube sampling over the space $[-b, b]^{n_p}$ instead). This transformation can be helpful in cases where the parameters are known to be positive and may vary in scale by several orders of magnitude. Combining LHS and the log transformed parameter space is often found to work quite well, although it can still perform poorly when the number of model parameters is large [17]. This technique can be used as the sampling algorithm for a multi-start method being applied to solve our original problem or in any of the approaches we describe next whenever sampling is required.

2.2 Methods for obtaining a suitable p_0

2.2.1 Using a Crude ODE solver

One way to generate p_0 is to use a crude ODE solver instead of the reliable CRK solver we use in the final optimization. For example, we might use a low order method with a fixed step size, such as Euler's method. Alternatively, if a reliable ODE solver with error control is required in order to simulate the IVP with any level of accuracy, we might choose a relaxed tolerance to reduce the computational effort being used to approximate the ODE.

2.2.2 Smooth and match estimator

It has been observed that the above mentioned approaches can still be computationally expensive. As we mentioned, the expensive part is often due to having to repeatedly simulate the underlying IVP numerically. Varah [49] and others [5, 20, 14] recognized that if one uses the observed values of $y(t)$ to approximate $y'(t)$, then one can formulate a related least squares problem,

$$\min_p \int_0^T \|(\tilde{y}'(t) - f(t, \tilde{y}(t), p))\|^2 \, dt, \tag{2.1}$$

where $\tilde{y}(t)$ is an approximation of $y(t)$ over the interval, $[0, T]$, based on the observed data. Note, in the work of Varah [49], a sum over the observation times was used, rather than an integral. An estimate for p obtained in this way is referred to as a smooth and match estimator (SME) [20]. Recently, a similar approach has appeared in the computational statistics and machine learning literature, which is referred to as Gradient Matching [34]. Another similar approach is used in large, underdetermined systems arising in systems biology [27]. Their approach differs in that they assume f is of the form,

$$f(t, y(t), p) = Sv(t, y(t), p), \tag{2.2}$$

where S is a stoichiometric matrix of constants and $v(t, y(t), p)$ is a rate vector. For their approach, one requires measurements of not only $y(t)$, but also $v(t)$.

In terms of computation, a major benefit of SME is that the numerical derivatives to be approximated can be significantly faster to compute than a simulation of an ODE. We also do not have to worry about what happens when a set of parameters would cause the simulation to fail. As with our original least squares problem, several techniques exist for performing the minimization in (2.1).

Note that while this approach cannot estimate initial conditions, we can try to estimate initial conditions directly from the observations. For example, on problems with some observations close to $t = 0$, we might extrapolate based on these observations. Alternatively, Dattner has suggested estimating initial conditions using the integral form of the ODE,

$$\min_{y_0} \int_0^T \left\| \tilde{y}(t) - \left(y_0 + \int_0^t f(s, \tilde{y}(s), p_{\text{sme}}) \, ds\right) \right\|^2 dt, \tag{2.3}$$

where p_{sme} are the initial estimates from (2.1) and y_0 are the initial conditions to be estimated. Note, this formulation is linear in y_0.

2.2.3 Numerical discretization-based estimation methods

Wu et al. [54] have proposed a similar family of methods, which they call numerical discretization-based estimation (DBE) methods. Briefly, these methods are based on one-step discretization methods of the form,

$$y(t_{i+1}) = y(t_i) + (t_{i+1} - t_i)F(t, y(t_i), y(t_{i+1}), p), \tag{2.4}$$

where $F(t, y(t_i), y(t_{i+1}), p)$ is determined by the discretization method. For example, if Euler's method is used,

$$F(t, y(t_i), y(t_{i+1}), p) = f(t_i, y(t_i), p). \tag{2.5}$$

The objective function for DBE is given by,

$$\min_p \sum_{i=1}^{n_o} \sum_{j=1}^{n_y} \Big(\frac{\hat{y}_j(t_{i+1}) - \hat{y}_j(t_i)}{t_{i+1} - t_i} - F_j(t, \hat{y}(t_i), \hat{y}(t_{i+1}), p) \Big)^2. \tag{2.6}$$

Note, in the case of Euler's method, this is precisely the divided differences approximation we have used in the past [10]. We also note that their work differs from ours in that we use IVP solvers with error control (variable step size), whereas they use fixed step size methods with no error control.

2.2.4 Choice of Smoother

To apply these smooth and match methods, we require a smooth estimate of the observed state variables. A variety of approaches have been used in similar applications. For example, cubic splines by Varah [49] and Bellman and Roth [5]. For problems where the data are not very dense and contain relatively small amounts of noise, cubic splines are a suitable choice. If the data are sufficiently dense, it may be better to use a bandwidth smoother, such as local polynomial estimation (LPE). This approach was used in [13]. If the data are sparse and noisy, penalized cubic splines may be more appropriate. We have used these various approaches in our numerical experiments, depending on the density of the observations and the amount of noise. One limitation of using a bandwidth smoother, like LPE, is that it requires the user to choose a suitable bandwidth. For example, if the data contain some regions with sharp peaks and other flat regions, then a balance must be found between oversmoothing the peaks and undersmoothing the flat regions. In [47], the authors investigate ways to get around this issue.

Local Polynomial Estimation

Given observations $\hat{y}(t_i)$, $i = 1, \ldots, n_o$, we seek a smoothed estimate, $\tilde{y}(t)$. To estimate the j^{th} component of $\tilde{y}(t)$, for a given t, we use local polynomial estimation of order l. This involves solving the weighted least squares (WLS) problem,

$$\tilde{v}_j(t) = \underset{v_j \in \mathcal{R}^{(l+1)}}{\arg\min} (Y_j - U v_j)^T W (Y_j - U v_j),$$

where,

$$v_j = \begin{bmatrix} \tilde{y}_j(t) & \tilde{y}_j'(t) b & \ldots & \tilde{y}_j^{(l)} b^l \end{bmatrix}^T,$$

$$U = \begin{bmatrix} 1 & \frac{t_1 - t}{b} & \ldots & \frac{(t_1 - t)^l}{b^l l!} \\ 1 & \frac{t_2 - t}{b} & \ldots & \frac{(t_2 - t)^l}{b^l l!} \\ & & \ldots & \\ 1 & \frac{t_{n_o} - t}{b} & \ldots & \frac{(t_{n_o} - t)^l}{b^l l!} \end{bmatrix},$$

Y_j is the vector of observed values of state j, b is the bandwidth of the kernel, and W is a diagonal matrix of weights, where $W_{ii} = K(\frac{t_i - t}{b})$, and $K(\cdot)$ is an appropriate kernel function. For example, as was done in [13], we use the Epanechnikov kernel,

$$K(t) = \begin{cases} \frac{3}{4}(1 - t^2) & \text{if } |t| \leq 1 \\ 0 & \text{otherwise} \end{cases}.$$

Note, we can also determine $\tilde{y}'(t)$, since $\frac{\tilde{v}(t) e_2}{b} = \tilde{y}'(t)$.

2.2.5 Limitation

A significant limitation of the above mentioned approaches is that they require data to be available for all components of $y(t)$. In some applications, the measured data are actually a function of $y(t)$ or only include a subset of the components of $y(t)$. In both of these cases, the approaches above are not directly applicable. In Chapter 5 we investigate ways to handle the case where only a subset of the components of $y(t)$ are observed. We now turn our attention to how the structure of the ODE can be used to reduce the computational cost of the first stage.

2.3 Exploiting ODE Structure

The structure of the system of ODEs can be used in two specific ways. First, some parameters may appear linearly in f. If we use the structure of the ODE assumed in (1.3), (2.1) becomes,

$$\min_q \int_0^T \|(\tilde{y}'(t) - G(t, \tilde{y}(t), q)r(q) - g(t, \tilde{y}(t), q)\|^2 \, dt. \tag{2.7}$$

Since r appears linearly in (2.1), we can view this as a minimization over only the nonlinear parameters, q. That is, for any given q, the best fit r is uniquely determined as,

$$r(q) = \left(\int_0^T G(t, \tilde{y}(t), q)^T G(t, \tilde{y}(t), q) \, dt \right)^{-1} \int_0^T G(t, \tilde{y}(t), q)^T (\tilde{y}'(t) - g(t, \tilde{y}(t), q)) \, dt, \tag{2.8}$$

which is the well known solution of a linear least squares (LS) problem. If we again use the structure of the ODE assumed in (1.3), (2.3) can be reformulated to take advantage of the fact that the integral form of the ODE is linear in both r and y_0. In this case, we have,

$$\min_q \int_0^T \left\| \tilde{y}(t) - \left(y_0 + \left[\int_0^t G(s, \tilde{y}(s), q) \, ds \right] r(q) + \int_0^t g(s, \tilde{y}(s), q) \, ds \right) \right\|^2 \, dt. \tag{2.9}$$

Since this least squares problem is linear in both r and y_0, the nonlinear optimization is only over q. Unlike SME, this does not actually require us to approximate $y'(t)$, since the IVP has been integrated. We will refer to this approach as integral smooth and match estimator (INT-SME).

Second, while the solution of a general system of ODEs will be coupled, it may be the case that the system of ODEs can be decoupled. That is, subsets of the parameters may appear in f, such that each subset can be independently determined. Similar remarks about the benefits of decoupling are discussed in [27]. To illustrate this decoupling, we consider the Mendes Problem, which we describe in more detail in Section 2.6.6. For now, we are simply interested in the structure of $G(t, y(t), q)$ for this model (note, $g(t, y(t), q) = 0$ for this model). The sparsity of $G(t, y(t), q)$ is shown in Figure 2.1. For this problem, we see that G consists of 7 blocks. It turns out that each of these blocks is also independent with respect to q, so the SME requires the approximate solution of 7 independent NLS

problems. This significantly reduces the computational burden of attempting any kind of global search over p, where $n_p = 36$ for this example. The 7 subproblems are summarized in Table 2.1. We see that the largest subproblem now requires a global search over only 6 nonlinear parameters, whereas if we only took advantage of the fact that some parameters are linear, we would have been attempting a global search over all 21 of the nonlinear parameters. If no structure was used, we would have to search over all 36 parameters.

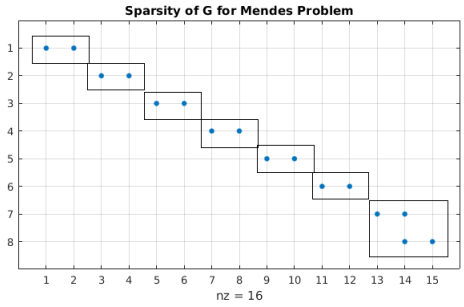


Figure 2.1: Each dot denotes a non-zero entry in G for the Mendes test problem. The matrix is sparse and contains 7 independent blocks.

Subproblem #	$\|r\|$	$\|q\|$	# of components of f
1	2	4	1
2-3	2	4	1
4-6	2	1	1
7	3	6	2
total	15	21	8

Table 2.1: Summary of the 7 subproblems for the Mendes Problem

2.3.1 B3 Test Problem Example

To further motivate the potential gains of exploiting structure, we consider a larger model from systems biology. This is one of six test problems fully specified in BioPreDyn-bench [50], which is a set of benchmark problems from systems biology. While this thesis focuses on parameter estimation problems where the parameters are identifiable, a common complication that often arises when estimating parameters in large models from systems biology is that the parameters are not all identifiable. No numerical experiments are performed for this problem, but we mention this problem simply to demonstrate how the structure of a model can still be used to divide the problem into subproblems.

The B3 test problem is a metabolic model of E. coli, consisting of 47 states and 178 parameters. Of these parameters, 82 are linear and 96 are nonlinear. The structure of

the model can be seen in Figure 2.2. The sizes of the subproblems are summarized in Table 2.2. We see that the first subproblem is quite large, due to four parameters that appear in almost all components of $f(t, y(t), p)$. If we assume fixed values for these four parameters, then the model can be broken into more manageable subproblems (Figure 2.3 and Table 2.3). Note, these four parameters actually appear together in one component of f without any other parameters, so it may be possible to estimate these four parameters from the available data to obtain appropriate values to fix them at. If this is done, we are left with several small subproblems, a couple medium size subproblems, and the largest subproblem is reduced from 91 down to 51 nonlinear parameters. Given the large number of parameters in this subproblem, identifiability issues would likely require more parameters to be fixed, allowing for potentially more decoupling. We now present an algorithmic description of our two stage procedure.

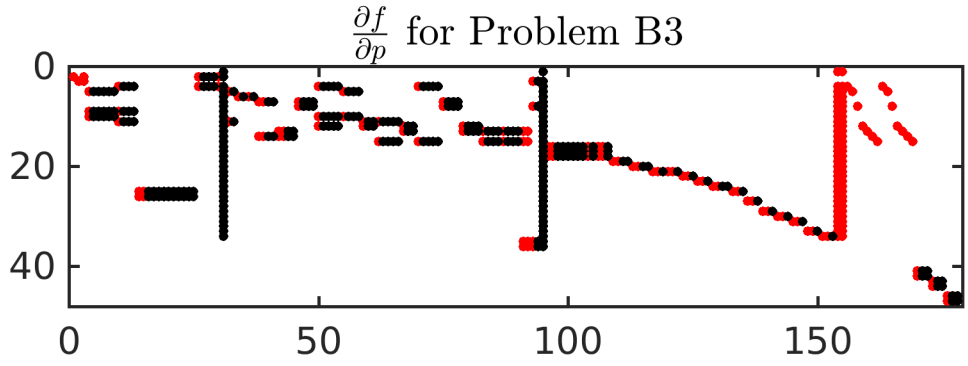


Figure 2.2: Sparsity pattern of the linear and nonlinear parameters for Problem B3. The (i,j) entry is coloured if $\frac{\partial f_i}{\partial p_j}$ is non-zero. The linear parameters are red and the nonlinear parameters are black.

Subproblem #	$\lvert r\rvert$	$\lvert q\rvert$	# of components of f
1	79	90	36
2-4	1	2	2
total	82	96	42

Table 2.2: Summary of the Subproblems for Problem B3

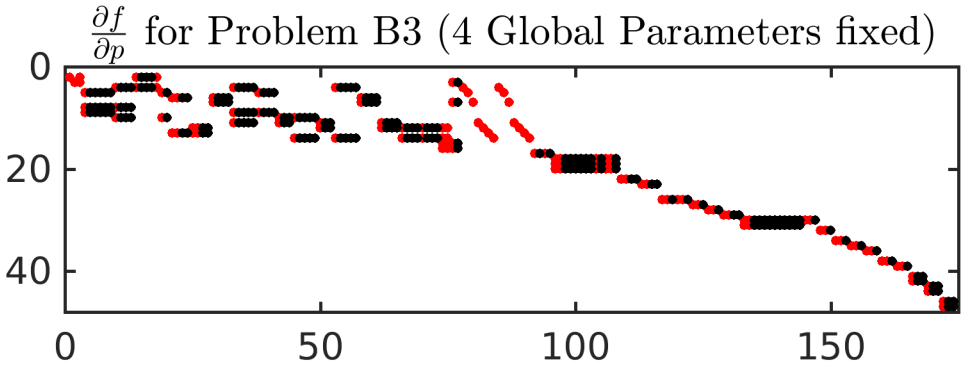


Figure 2.3: Sparsity pattern of the linear and nonlinear parameters for Problem B3 with decoupling. The (i,j) entry is coloured if $\frac{\partial f_i}{\partial p_j}$ is non-zero. The linear parameters are red and the nonlinear parameters are black.

Subproblem #	$\|r\|$	$\|q\|$	# of components of f
1	40	51	15
2,4,5,9	2	2	1
3	5	8	3
6	4	2	1
7,8,11-16	2	1	1
10	4	11	2
17-19	1	2	2
total	80	94	39

Table 2.3: Summary of the Subproblems for Problem B3 when we fix the four parameters coupling the first subproblem

2.4 Full Procedure

We assume that, for the initial procedure being used, the parameters can be split into K independent sets of parameters. Obtaining an initial guess for the k^{th} set of parameters, p^k, will be referred to as subproblem S_k. We refer to the nonlinear parameters in each set as q^k. S_k may be any one of the initial guess generation procedures previously described. The full procedure is given in Algorithm 1.

We found that $N_{\text{best}} = 10$ and $N_{\text{repeat}} = 3$ seemed to give a reasonable balance between speed and robustness. We also tried various values for N^k_{samples} and found that choosing N^k_{samples} proportional to the square of the number of nonlinear parameters in subproblem S_k seemed to work well for the test problems considered in this thesis.

Stage I:

foreach *subproblem* S_k, $k = 1, \ldots, K$ **do**

obtain N^k_{samples} samples of q^k from the log of the parameter space using LHS

calculate the subproblem objective function (and hence optimal r^k) for each sample and sort based on this

run local optimizer from the N_{best} samples of q^k or fewer if the same minimizer is found N_{repeat} times

let P_k be the set of unique minimizers found for subproblem S_k

end

let $P_0 = P_1 \times P_2 \times ... \times P_K$

compute the objective function for each $p_0 \in P_0$ and sort based on this

Stage II:

run local optimizer on p_0's from P_0 until satisfied that best fit parameters have been found

Algorithm 1: Two Stage Parameter Estimation Procedure

2.5 Related Methods

We now briefly discuss several methods that are related to our two stage procedure.

2.5.1 Accelerated Least Squares

The accelerated least squares (ACCEL) procedure proposed by Dattner [13] also consists of two stages. The first stage can either be the SME or INT-SME previously mentioned. The second stage is to perform a single iteration of Newton's method, from this initial estimate for the parameters, to minimize the objective function (1.7). Under certain conditions, one is guaranteed that the estimated parameters obtained in this way will be statistically the same as if one were to explicitly minimize (1.7). While this is somewhat expensive, as it requires computation of the matrix of second partials of (1.7), it only needs to be done once and, in the case of ODEs, it is straightforward to approximate these partials. ACCEL closely resembles our proposed procedure, but differs in how we perform the final estimation, which we discuss in detail in the next chapter, and in the fact that we explicitly exploit the structure of the system of ODEs in the first stage.

2.5.2 Multiple Shooting

In this thesis we only consider simple shooting in our numerical experiments, but it is worth mentioning multiple shooting briefly. Multiple shooting is a more robust form of

simple shooting. This method is widely used for the numerical solution of boundary value problems in ODEs (see, for example [29, 9]). It has also been suggested for estimating the parameters in systems of ODEs [48, 39].

In multiple shooting, the interval over which the system is simulated is divided into N_{MS} subintervals, where the first interval is $I_0 = [0, \tilde{t}_1]$, the r^{th} interval is $I_r = [\tilde{t}_r, \tilde{t}_{r+1}]$, and $I_{N_{MS}-1} = [\tilde{t}_{N_{MS}-1}, T]$. Additional parameters are added to specify the state vector at the beginning of each subinterval after I_0 (assuming initial conditions are already included as parameters). This means we have $(N_{MS} - 1)n_y$ additional parameters to estimate. Finally, we introduce equality constraints at the boundary of each interval,

$$y(\tilde{t}_r, p; \tilde{t}_{r+1}) = y(\tilde{t}_{r+1}) \ \text{, for } r = 0, \ldots, N_{MS} - 2, \tag{2.10}$$

where $y(\tilde{t}_r, p; \tilde{t}_{r+1})$ is the value of the state vector at the right end point of I_r, and $y(\tilde{t}_{r+1})$ is the value of the state vector at the left end point, which are the additional parameters to be estimated. Note, these additional constraints are added to the optimization problem (1.7).

One advantage of this approach is that it allows discontinuities in the intermediate trajectories (i.e. violation of the introduced equality constraints) to exist during the optimization. Also, since it restarts the simulation at the start of each subinterval, it is less likely that the IVP solver will fail. The presence of the additional equality constraints ensures that the trajectory at convergence will be continuous almost everywhere. When used in conjunction with a gradient based optimizer, a downside of multiple shooting is that each iteration is generally more computationally expensive than simple shooting. We note that the simulations on each subinterval can be done in parallel on each iteration. Multiple shooting was extended in [23] to a class of DDEs with constant delays.

2.5.3 Method of Ramsay

As we mentioned in Section 1.2.3, there is another way to approach solving the NLS problem (1.7). In the approach of Ramsay [40], the ODE constraint is treated by the inclusion of a penalty term in the objective function (1.7). The state vector is approximated by $x(t)$, which is represented using B-splines,

$$x_i(t) = \sum_{k=1}^{N_B} c_{ik} B_{ik}(t) = c_i^T B_i(t), \tag{2.11}$$

where N_B is the number of basis functions used and c_i is the vector of coefficients intro-

duced to parameterize $x_i(t)$. The penalty term for the i^{th} component is given by,

$$PEN_i(x) = \int_0^T \lambda_i (x_i'(t) - f_i(t, x, p))^2 dt, \tag{2.12}$$

where λ_i determines how much weight the i^{th} penalty term has. For simplicity, we'll assume each $\lambda_i = \lambda$. The full objective function is then,

$$\sum_{i=1}^{n_o} \sum_{j=1}^{n_y} \frac{(\hat{y}_j(t_i) - x_j(t_i, p))^2}{2} + \sum_{j=1}^{n_y} PEN_j(x) \tag{2.13}$$

Since an ODE solver is not being used to evaluate the objective function, the state vector is allowed to violate the ODE constraint on intermediate iterations of the optimization, similar to multiple shooting. Also, there is no requirement to explicitly simulate the ODE across the interval of interest. Instead, each time point of interest may be handled simultaneously. In [40], the authors go on to discuss how to use parameter cascades to efficiently solve this optimization problem. We note that their approach requires several additional partial derivative expressions to be provided by the user and rather complicated expressions arise.

If the B-splines used are not able to satisfy the ODE to sufficient accuracy, then the parameter estimates may become skewed, depending on the value of λ chosen, as shown in Figure 2.4. Note, this will also depend on the quadrature rule used to estimate the integral in the penalty term. For these figures, we estimated the parameters for the Barnes problem, with $n_o = 40$, a B-spline knot placed at every second observation, and the trapezoidal rule used to approximate the penalty term with a spacing of 0.05.

We see that with this setup, the choice of λ can significantly alter the final fit and estimated parameter values. When λ is chosen too small, the ODE constraint is not being satisfied and the data fitting part of the objective function dominates. When λ is chosen too large, the ODE constraint begins to skew the final fit. Ideally, the spline would be flexible enough to exactly match the ODE and the quadrature rule used accurately approximates the penalty term. This would result in the penalty term not contributing to the objective function at convergence. However, in our example, the spline is unable to match the ODE and instead influences the parameter estimates.

As this example demonstrated, it is essential that an appropriate smoothing parameter, λ, be chosen, as well as an appropriate mesh for the B-splines being used to approximate the solution of the ODE. If the solution of the ODE varies rapidly at some points and is flat in others, a non-uniform mesh should be used. One could use the observed data to estimate the rate of change in the trajectory to inform the choice of the

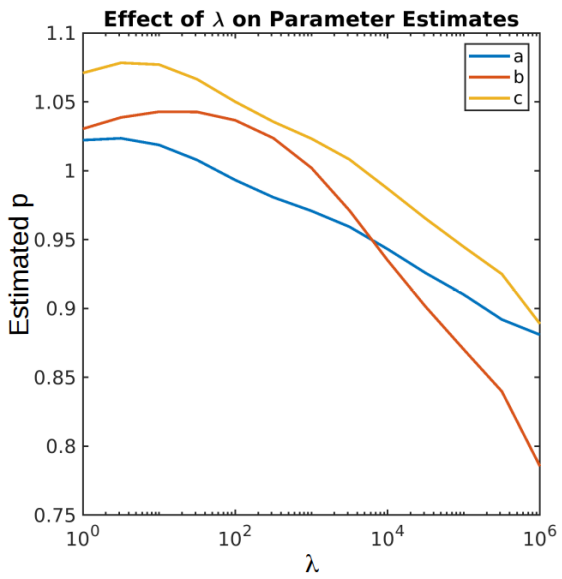


Figure 2.4: Parameter estimates obtained using Ramsay's approach for the Barnes problem, for varying values of the smoothing parameter, λ. The true parameter vector is $[1, 1, 1]$.

B-spline mesh. We note that by using an ODE solver with error control and a continuous interpolant, we are constructing such a non-uniform mesh, with confidence that the ODE is being satisfied over the interval. Thus, we don't have to worry about this difficulty.

Lastly, it is interesting to note that we can easily adapt Ramsay's approach to view it as an approach for obtaining p_0. If we solve (2.13) with $\lambda = 0$, then this means that we are fitting the B-splines to the observed data. If we then fix the coefficients of the B-splines, the first term in (2.13) becomes a constant and the second term is precisely (2.1), with $x(t)$ playing the role of $\tilde{y}(t)$. So, applying Ramsay's approach in this way is equivalent to applying SME, with B-splines used to smooth the data.

2.6 Numerical Experiments

We now present results for several numerical examples to demonstrate the performance of the methods we have discussed.

2.6.1 Comparison of approaches to obtain p_0

To compare the performance of the approaches discussed earlier for obtaining a suitable p_0, we perform the following experiment using the FitzHugh-Nagumo model.

FitzHugh-Nagumo Model

The FitzHugh-Nagumo equations model potentials of squid neurons [16]. The model is given by,

$$y_1'(t) = c\left(y_1(t) - \frac{y_1(t)^3}{3} + y_2(t)\right), \tag{2.14}$$

$$y_2'(t) = \frac{-\left(y_1(t) - a + by_2(t)\right)}{c}, \tag{2.15}$$

where $y_1(t)$ is a voltage across the neuron and $y_2(t)$ is a response current, inversely related to $y_1(t)$. The initial conditions and parameters are chosen to be $y_1(0) = -1$, $y_2(0) = 1$, $a = 0.2$, $b = 0.2$, and $c = 3.5$, for $t \in [0, 20]$. This parameterization of the model is considered a challenging test problem due to the observation that the corresponding objective function has many local minima [51]. The true trajectories corresponding to these parameters are shown in Figure 2.5. Both the parameters and initial conditions are estimated in this problem.

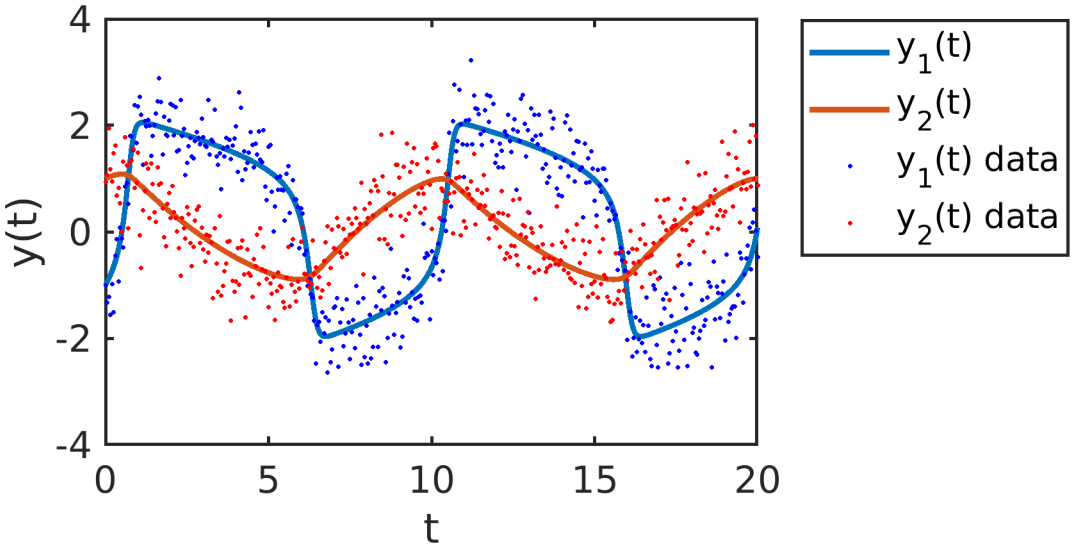


Figure 2.5: True trajectories for FitzHugh-Nagumo model, along with a sample of noisy data with $\sigma = 0.5$.

For the purposes of this experiment, we have fixed the nonlinear parameter, c, so we are only estimating the two linear parameters, a and b. This allows us to more easily visualize the quality of the p_0's being generated. As in [40], we assume both state variables are observed every 0.05 time units over the interval $[0, 20]$. This results in 400 observations. The true trajectory is generated using an accurate ODE solver and noise with a standard deviation of 0.5 is added at each observation time. This is

done multiple times to give us 200 sets of noisy observations. For each data set, several approaches are used to generate p_0. For INT-SME and SME, an LPE smoother is used with a reasonable bandwidth chosen based on the number of data points. For Euler, the solver uses a fixed step size of 0.05 to simulate the IVP. The results are summarized in Figure 2.6 and Table 2.4. We see that all the approaches are able to get us sufficiently close to the true parameter values, so that a quasi-Newton optimizer was able to rapidly converge. However, the quality of the p_0's do vary between the methods. For example, when we use Euler's method to crudely solve the IVP, we see that the guesses don't vary as much between data sets, but they are very clearly skewed. Furthermore, using Euler's method will be more expensive than using the other three approaches, since it is a nonlinear optimization, whereas the other approaches are all linear optimizations. INT-SME, SME, and DBE all perform similarly, but INT-SME provides noticably tighter guesses, which may save one or two iterations on the final optimization. Since each iteration of the final optimization is generally much more expensive than INT-SME, it is expected that the additional cost of INT-SME compared to DBE may be worth it. DBE performs surprisingly well here, since others have noted that divided difference based estimates of the derivative perform poorly when there is a high level of noise. We did find that if forward differences (FD) was used, the results were quite poor, so we have used centered differences (CD) to obtain these results.

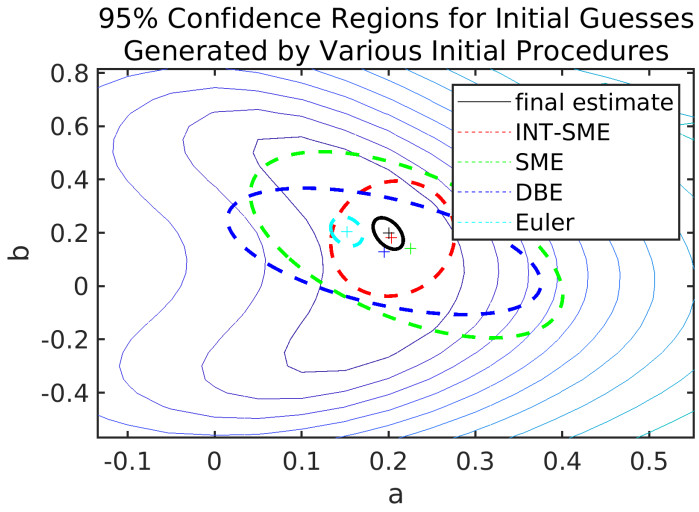


Figure 2.6: Comparison of p_0's generated by various approaches for the FitzHugh-Nagumo example. The 95% confidence ellipses are based on the mean and covariance of the p_0's and $\hat{p}$'s generated for each of the 200 simulated data sets. The confidence ellipses are overlaid over a contour map of the full objective function.

method	p_0 cost (s)
SME	0.081
INT-SME	0.084
DBE	0.012
Euler [1]	-

Table 2.4: Cost in seconds of procedures for generating p_0 for the FitzHugh-Nagumo example. DBE is roughly 7 times faster than INT-SME and SME, since it doesn't require any smoothing of the noisy data. This experiment was run on an Intel X5675 (3.08 GHz).

σ	n_o	SME	INT-SME	DBE	Euler
0.1	25	1.0e-01 (5.4e-02)	1.2e-01 (5.9e-02)	6.6e-02 (3.9e-02)	1.3e-01 (2.1e-02)
0.1	100	7.1e-02 (3.9e-02)	6.2e-02 (3.3e-02)	4.1e-02 (2.5e-02)	6.9e-02 (7.6e-03)
0.1	400	5.5e-02 (2.7e-02)	2.3e-02 (1.4e-02)	2.7e-02 (1.5e-02)	4.9e-02 (1.6e-03)
0.5	25	3.6e-01 (2.4e-01)	3.4e-01 (2.3e-01)	2.6e-01 (1.4e-01)	1.5e-01 (8.1e-02)
0.5	100	2.3e-01 (1.6e-01)	1.6e-01 (1.2e-01)	1.6e-01 (8.9e-02)	8.2e-02 (3.5e-02)
0.5	400	1.4e-01 (8.6e-02)	7.7e-02 (4.8e-02)	1.1e-01 (6.3e-02)	5.4e-02 (9.5e-03)

Table 2.5: The average distance between p_0's and the true parameter vector for varying levels of noise and number of observations, with standard deviation in brackets, for the FitzHugh-Nagumo example.

We also further investigate how these approaches perform when we vary the noise level and number of observations. These results are shown in Figure 2.7 and Table 2.5. From Figure 2.7, we observe that when there is less noise, the initial guesses are closer to the true parameter values for all methods except Euler. In the case of Euler, the variance of the estimates is reduced, but the bias in the estimate is not reduced. In Table 2.5, we report the average distance from the initial guesses generated to the true parameter vector. We see that as the number of observations decreases, all methods perform worse, although SME and INT-SME seem to be more affected by a small number of observations than DBE and Euler. This seems to be caused by the reduced number of observations available to the bandwidth smoother.

Cost of Final Estimation from the generated p_0's

We also report how long the final optimization takes from some of the initial guesses. As we noted, all p_0's generated lie within the basin of attraction of the true parameters, so the final estimation converges to the global minimum. The results are shown in Table

[1] Since Euler results in a nonlinear optimization problem, the actual cost is dependent on how the problem is specified. To generate these results, we started the local minimizer sufficiently close to the minimizer to obtain fast convergence, so we don't report timing results for this method.

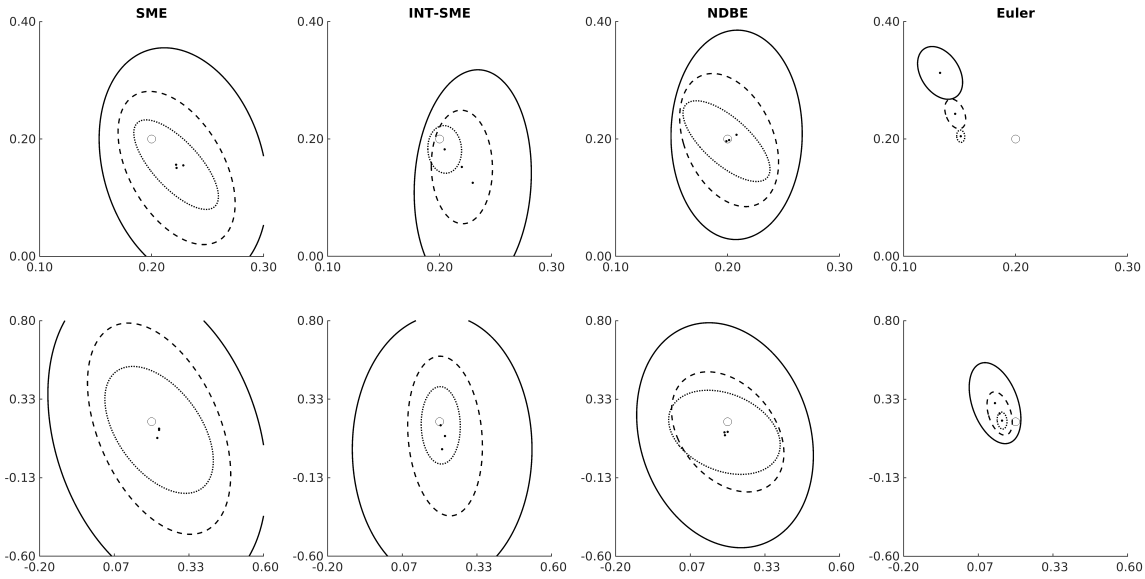


Figure 2.7: Comparison of p_0's generated by various approaches for varying level of noise and number of observations for the FitzHugh-Nagumo example. The 95% confidence ellipses are based on the means and covariances of the p_0's generated for each of the 200 simulated data sets. The true parameter vector of $[0.2, 0.2]$ is denoted by a circle and the mean guesses by solid dots (centers of ellipses). Note, the scale is different between the two rows. (25 observations (solid line), 100 observations (dashed line), 400 observations (dotted line), top row is $\sigma = 0.1$, bottom row is $\sigma = 0.5$)

2.6. As expected, all methods perform about the same, although Euler consistently requires exactly four iterations in the case of low noise and a large number of observations available, since there is so little variance in the p_0 generated by Euler in that case. In the case of more noise and fewer observations, DBE and Euler require around 1 iteration less than SME and INT-SME on average.

As we saw in the timing comparison in Table 2.4, all of the approaches cost a similar amount, although DBE is somewhat cheaper since it is neither simulating the IVP nor using a smoother. However, we note that in the case where some parameters appear nonlinearly, some kind of global optimization strategy must be used. In such a case, the relative cost of running the smoother becomes much less important, since the smoothing of the data only has to be done once. Ignoring the fixed cost of smoothing the data, the cost of evaluating SME, INT-SME, and DBE is practically the same. This is not the case with Euler, since evaluation of its objective function is inherently more expensive. For the FitzHugh-Nagumo example considered here, one iteration of the final optimization takes about 0.1s. This means that all of these initial procedures only cost a fraction of an iteration of the final optimization, with DBE only costing about 10%.

σ	n_o	SME	INT-SME	DBE	Euler
0.1	25	0.44 (0.09)	0.46 (0.09)	0.41 (0.09)	0.43 (0.07)
0.1	400	0.36 (0.05)	0.33 (0.04)	0.34 (0.02)	0.43 (0.004)
0.5	25	0.82 (0.39)	0.88 (0.50)	0.70 (0.34)	0.68 (0.44)
0.5	400	0.43 (0.12)	0.38 (0.09)	0.41 (0.07)	0.38 (0.05)

Table 2.6: The time taken to perform the final optimization for the FitzHugh-Nagumo example. We report the average time taken from the guesses generated by each approach for 50 of the 200 simulated data sets (standard deviation in brackets). One iteration of the final optimization takes about 0.1 seconds.

2.6.2 Effect of the Number of Observations on Initial Procedures

To further investigate how the initial procedures behave when the number of observations is varied, we consider the Barnes problem.

Barnes Problem

The Barnes Problem is often used in the parameter estimation literature for ODE models [51, 49]. It refers to a specific parameterization of the predator-prey model, given by,

$$y_1{}'(t) = ay_1(t) - by_1(t)y_2(t), \tag{2.16}$$

$$y_2{}'(t) = by_1(t)y_2(t) - cy_2(t), \tag{2.17}$$

where $y_1(t)$ is the population of predators, $y_2(t)$ is the population of prey. The true parameters and initial conditions are specified to be $a = 1$, $b = 1$, $c = 1$, $y_1(0) = 1$, and $y_2(0) = 0.3$, for $t \in [0, 20]$. The true trajectories corresponding to these parameters are shown in Figure 2.8. Both the parameters and initial conditions are estimated in this problem.

For this experiment, we fix the noise level at $\sigma = 0.1$. A uniform mesh of observation points between $t = 0$ and $t = 20$ is used, but we vary the spacing of the mesh. The finest mesh has a spacing of 0.25 and we use meshes with integer multiples of this spacing, up to a coarsest spacing of 2.5. The results are summarized in Table 2.7, where we report the mean bias for each of the three methods for ten mesh spacings. For INT-SME, we report results where no smoothing was done (results when a smoother is used look similar). For SME, an LPE smoother was used. For DBE, we report results for two different centered

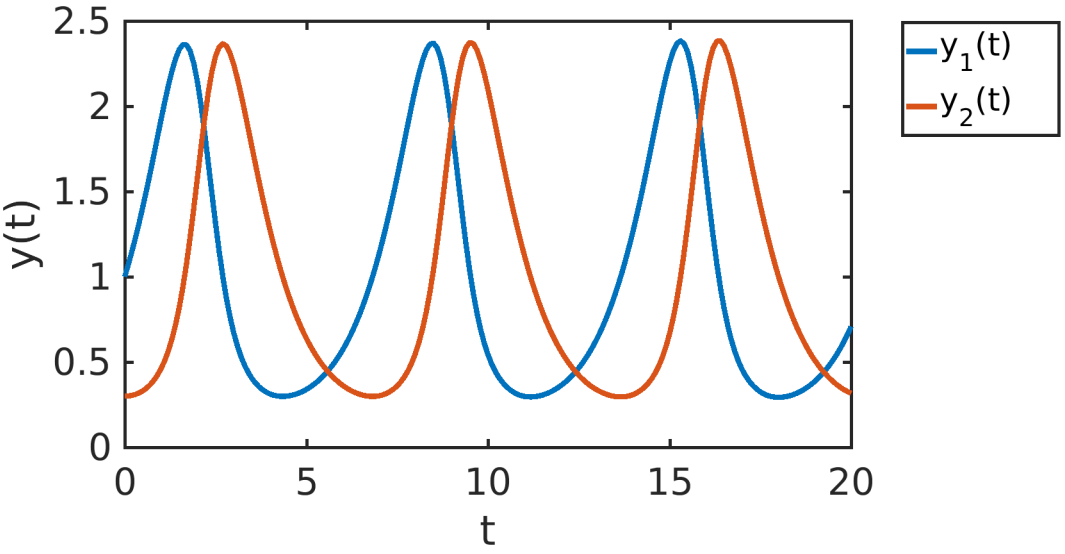


Figure 2.8: True trajectories for the Barnes Problem

discretizations to approximate the derivative,

$$y'(t_i) \approx f(t_i, \bar{y}(t_i), p) \approx \frac{\bar{y}(t_{i+1}) - \bar{y}(t_{i-1})}{(t_{i+1} - t_{i-1})},\ i = 2, \ldots, n_o - 1, \tag{2.18}$$

or,

$$y'(\frac{t_i + t_{i+1}}{2}) \approx f(\frac{t_i + t_{i+1}}{2}, \frac{\bar{y}(t_i) + \bar{y}(t_{i+1})}{2}, p) \approx \frac{\bar{y}(t_{i+1}) - \bar{y}(t_i)}{(t_{i+1} - t_i)},\ i = 1, \ldots, n_o - 1. \tag{2.19}$$

For this example, we see that (2.19) gives better estimates than (2.18). As expected, all methods perform well when there are a large number of observations available, with the estimates becoming poor as the number of observations is reduced. We note that in terms of computational cost, SME, INT-SME, DBE take roughly 0.12s, 0.059s, and 0.017s, respectively.

Lastly, we note that not only is the number of observations important, but also the placement of the observations. We see that even though spacings of 2.5 and 2.25 result in $n_o = 8$, we get very different estimates in the two cases. This is due to the periodic nature of the trajectories of the Barnes IVP and where the observations end up being located along the oscillating trajectories.

For this test problem, the initial conditions are also parameters. For the SME and DBE procedures, initial conditions are estimated to be the observed values at $t = 0$. Given the noise level used, this results in asymptotic estimates of $y_1(0) = 1(0.1)$ and

dt	n_o	SME	INT-SME	CD-DBE(2.18)	CD-DBE(2.19)
0.25	80	-0.030 (0.021)	-0.006 (0.019)	-0.057 (0.020)	-0.010 (0.021)
0.5	40	-0.036 (0.031)	0.003 (0.030)	-0.126 (0.027)	0.010 (0.030)
0.75	26	-0.081 (0.030)	0.020 (0.033)	-0.256 (0.027)	0.026 (0.030)
1	20	-0.133 (0.034)	0.028 (0.046)	-0.390 (0.030)	0.056 (0.035)
1.25	16	-0.150 (0.031)	0.032 (0.080)	-0.466 (0.028)	0.101 (0.035)
1.5	13	-0.196 (0.038)	-0.147 (0.127)	-0.587 (0.032)	0.143 (0.042)
1.75	11	-0.303 (0.029)	-0.173 (0.111)	-0.678 (0.019)	0.213 (0.041)
2	10	-0.395 (0.043)	-0.634 (0.086)	-0.772 (0.023)	0.213 (0.061)
2.25	8	-0.367 (0.048)	0.040 (0.117)	-0.647 (0.019)	0.850 (0.099)
2.5	8	-0.556 (0.049)	-0.822 (0.054)	-0.879 (0.020)	0.126 (0.102)

Table 2.7: Mean bias of the 3 non-initial condition parameters for the Barnes test problem, for varying numbers of observations (standard deviation in brackets). This is based on 100 simulated data sets.

dt	n_o	$Y_1(0)$	$Y_2(0)$
0.25	80	0.994 (0.134)	0.293 (0.109)
0.5	40	1.020 (0.182)	0.277 (0.139)
0.75	26	1.008 (0.219)	0.283 (0.174)
1	20	1.009 (0.247)	0.276 (0.198)
1.25	16	1.048 (0.285)	0.360 (0.249)
1.5	13	0.896 (0.225)	0.567 (0.188)
1.75	11	0.909 (0.179)	0.536 (0.212)
2	10	1.406 (0.122)	0.408 (0.131)
2.25	8	1.396 (0.214)	0.706 (0.216)
2.5	8	1.386 (0.081)	0.892 (0.091)

Table 2.8: Mean estimate of initial condition parameters for the Barnes test problem using INT-SME, for varying numbers of observations (standard deviation in brackets). This is based on 100 simulated data sets. As reference, $y_1(0) = 1(0.1)$ and $y_2(0) = 0.29(0.08)$ is the estimate obtained from the simulated data.

$y_2(0) = 0.3(0.1)$. For INT-SME, the estimates of the initial conditions are given in Table 2.8. These results suggest that initial conditions should be estimated from the data directly, rather than by INT-SME, at least with this level of noise.

Cost of Methods for Barnes example

In terms of computational cost, we consider both the cost of generating p_0 and the cost of performing the final estimation. The results are summarized in Table 2.9. We see that INT-SME and DBE(2.18) take a bit longer in the final optimization. For INT-SME, this can be attributed to the fact that its initial guess for the initial conditions is not as reliable as the other methods. For DBE(2.18), this is probably due to the worse

initial guess for the other 3 parameters. SME and DBE(2.19) perform about the same in the final estimation, but the cost of the smoother in SME makes it the most expensive approach in this example. Note, these timing results are for 50 simulated data sets. All 4 approaches converged to the global minimum on 49 of the 50 data sets. Interestingly, the data set that INT-SME found a local minimizer was not the same as the data set the other 3 approaches all converged to a local minimizer. The local minimizer found corresponds to another periodic solution of lower frequency in both cases.

	SME	INT-SME	DBE(2.18)	DBE(2.19)
Initial (s)	0.118	0.059	0.017	0.018
Final (s)	0.200	0.235	0.255	0.195
Total (s)	0.318	0.294	0.272	0.213

Table 2.9: Cost in seconds of procedures for generating p_0 for the Barnes problem. This is for 50 of the simulated data sets from the case of $n_o = 80$.

2.6.3 Calcium Ion Example

This system of ODEs describes the oscillations of Ca^{2+} ions in the cytoplasm of eukaryotic cells, which play a role in cellular information processing. For a complete description of this model, see [30] where this model was first proposed. The model, as originally specified, is given by,

$$G^*{}_\alpha{}' = k_1 + k_2 G^*{}_\alpha - k_3 PLC^* \frac{G^*{}_\alpha}{G^*{}_\alpha + Km_1} - k_4 Ca_{cyt} \frac{G^*{}_\alpha}{G^*{}_\alpha + Km_2}, \tag{2.20}$$

$$PLC^{*\prime} = k_5 G^*{}_\alpha - k_6 \frac{PLC^*}{PLC^* + Km_3}, \tag{2.21}$$

$$\begin{aligned} Ca_{cyt}{}' = {} & k_7 PLC^* Ca_{cyt} \frac{Ca_{er}}{Ca_{er} + Km_4} + k_8 PLC^* + k_9 G^*{}_\alpha \\ & - k_{10} \frac{Ca_{cyt}}{Ca_{cyt} + Km_5} - k_{11} \frac{Ca_{cyt}}{Ca_{cyt} + Km_6}, \end{aligned} \tag{2.22}$$

$$Ca_{er}{}' = -k_7 PLC^* Ca_{cyt} \frac{Ca_{er}}{Ca_{er} + Km_4} + k_{11} \frac{Ca_{cyt}}{Ca_{cyt} + Km_6}, \tag{2.23}$$

where the state variables are concentrations of four compounds, which interact in the calcium-signaling pathway. In our notation, $[y_1\ y_2\ y_3\ y_4] = [G^*{}_\alpha\ PLC^*\ Ca_{cyt}\ Ca_{er}]$. Parameters are chosen to be $k_1 = 0.09$, $k_2 = 2$, $k_3 = 1.27$, $k_4 = 3.73$, $k_5 = 1.27$, $k_6 = 32.24$, $k_7 = 2$, $k_8 = 0.05$, $k_9 = 13.58$, $k_{10} = 153$, $k_{11} = 4.85$,$Km_1 = 0.19$,$Km_2 = 0.73$,$Km_3 = 29.09$,$Km_4 = 2.67$,$Km_5 = 0.16$,$Km_6 = 0.05$. Initial conditions are treated as known, and given by $y_1(0) = 0.12$, $y_2(0) = 0.31$, $y_3(0) = 0.0058$, and $y_4(0) = 4.3$. The model

is simulated for $t \in [0, 20]$. For this specific parameterization, the solution exhibits a limit cycle [39]. The true trajectories corresponding to these parameters are shown in Figure 2.9. Usually the nonlinear parameters are considered as fixed and only the linear parameters are estimated for this problem.

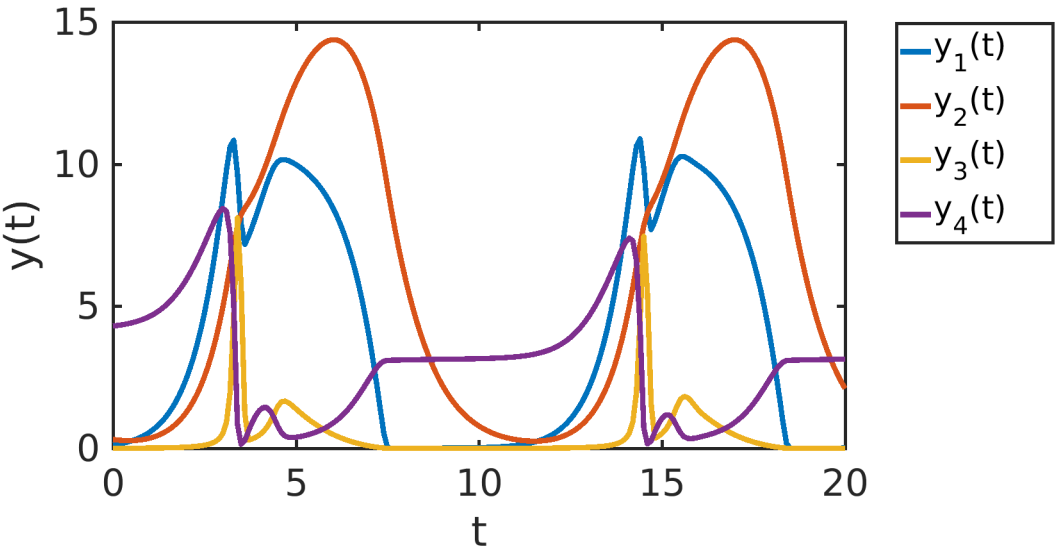


Figure 2.9: True trajectories for the Calcium Ion test problem

This model contains 11 linear parameters to be estimated, with 6 nonlinear parameters whose values are held fixed. For this experiment, we take observations every 0.1 time units, from $t = 0$ to $t = 10$. Noise is added relative to the magnitude of each component of the state vector, such that each observation has roughly 6.5% error. The results are shown in Figure 2.10. We only show results for SME and INT-SME, as the results of SME and DBE are very similar. For SME and INT-SME, we do not use a smoother in this example. We report the initial guesses generated by each method for each of the 11 parameters for 200 sets of simulated data. Both methods generate good guesses for most parameters, but they both struggle somewhat with the first and eighth parameters, while SME also produces guesses for parameters 9 and 10 that are consistently about half of their true value. On average, SME takes 0.0021s and INT-SME takes 0.0032s for this example.

As noted in [21], the choice of smoother can bias the estimates generated by procedures like SME. This is the case in this example, due to the sharp peaks in the trajectory. This can be addressed to some degree by attempting to put weights on the observations. One way to think of this is that since we only have a small number of samples near the peaks, they need to be given more weight if we hope to fit them. For example, we tried putting

more weight on the observations at times corresponding to the peaks and were able to reduce the bias in the estimates of parameters 9 and 10, but the estimates for parameter 11 became worse (results not shown) and the variance in the estimates of parameters 9 and 10 increased.

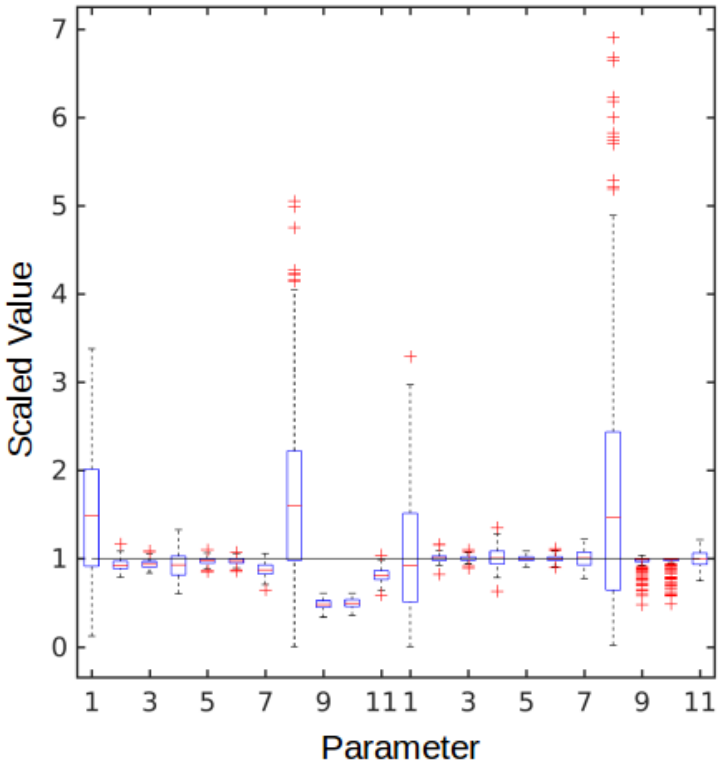


Figure 2.10: Boxplot of p_0's obtained for the Calcium Ion test problem using SME(left) and INT-SME(right) for 200 simulated data sets. Parameters have all been rescaled to be equal to one in this plot.

Given these initial guesses, we then attempted to perform the final estimation procedure for 50 of the 200 data sets. For these data sets, we note that the median objective function value at the generated initial guesses was around 670 for INT-SME and 2240 for SME.

Levenberg-Marquardt is used to perform the optimization, as implemented in lsqnonlin. We used the DDEM IVP solver to simulate the model trajectories and we used centered divided differences to approximate the required sensitivities, as described in Section 3.2. Using the initial guesses generated by SME and INT-SME, the final optimization converged to the global minimum in all cases, except once for SME, where a local minimum was found. The cost of the optimizations are summarized in Table 2.10. We see that SME took significantly more time to converge than INT-SME. The average time taken by INT-SME was 2.05s, while it was 3.91s for SME.

time	<1 s	<2 s	<3 s	<4 s	<5 s	≥5 s
SME	0	3	7	20	10	10
INT-SME	18	9	12	6	3	2

Table 2.10: Summary of time taken to converge to the global minimizer for the Calcium Ion test problem, using the p_0's generated by SME and INT-SME for 50 sets of simulated data.

2.6.4 DDE Examples

We now consider how these approaches perform on two constant lag DDE models.

Hutchinson's model

This is a DDE model of population dynamics, given by,

$$y'(t) = ry(t)\left(1 - \frac{y(t-\tau)}{K}\right), \tag{2.24}$$

where r is the growth rate, K is the carrying capacity, $y(t)$ is the population, and τ is the time lag. This model was introduced in [24] and is commonly studied in ecology. We use parameters similar to those used in [52]. The values are $r = 1$, $\tau = 1.9$, $K = 2000$, and $y(t) = y(0) = 1000$, for $t \leq 0$. The true trajectory corresponding to these parameters is shown in Figure 2.11. The parameters and initial condition are estimated in this problem.

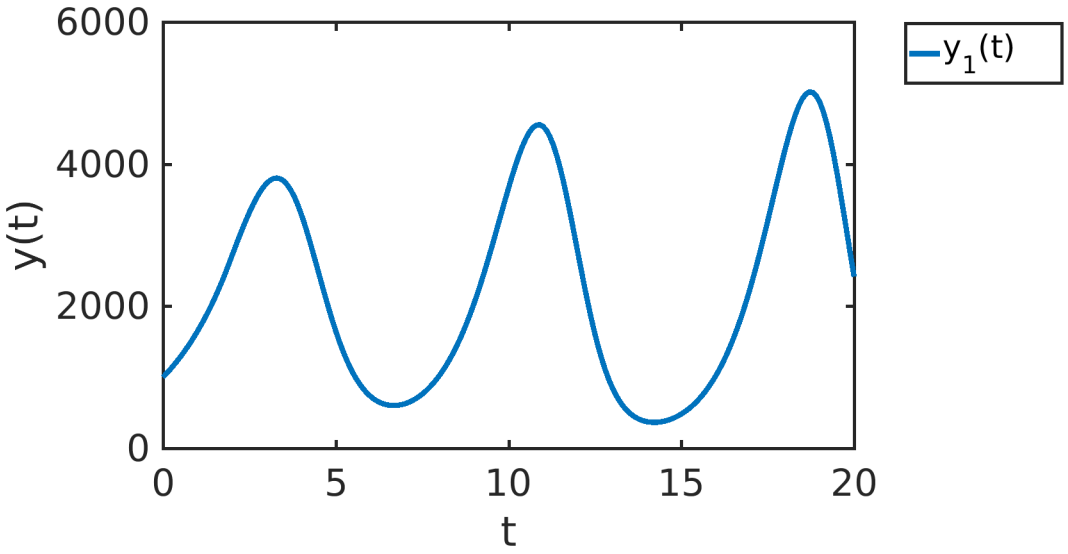


Figure 2.11: True trajectory for Hutchinson's model

The state vector is observed every time unit over the interval $[0, 20]$. This results in

21 observations. The true trajectory is generated using an accurate DDE solver and noise with a standard deviation of 10% of the average value of the state vector is added at each observation time. This is done multiple times to give us 50 sets of noisy observations. For each data set, DBE (with formula (2.19)) and INT-SME (without any smoother) are used to generate p_0. The results are shown in Figure 2.12. These p_0's are then used to obtain the final parameter estimates. The final optimization is performed using LM, as implemented in lsqnonlin, with the required sensitivities approximated using forward divided differences. Both DBE and INT-SME are able to give us sufficiently good p_0's to allow LM to quickly converge to the best fit parameters. We note that DBE tends to underestimate the lag parameter, τ, while INT-SME has significant variance on its initial guess for $y(0)$. In terms of timing, generating p_0 took 0.014s and 0.019s for DBE and INT-SME, respectively. The total time taken by each approach was 1.42s and 1.22s, on average.

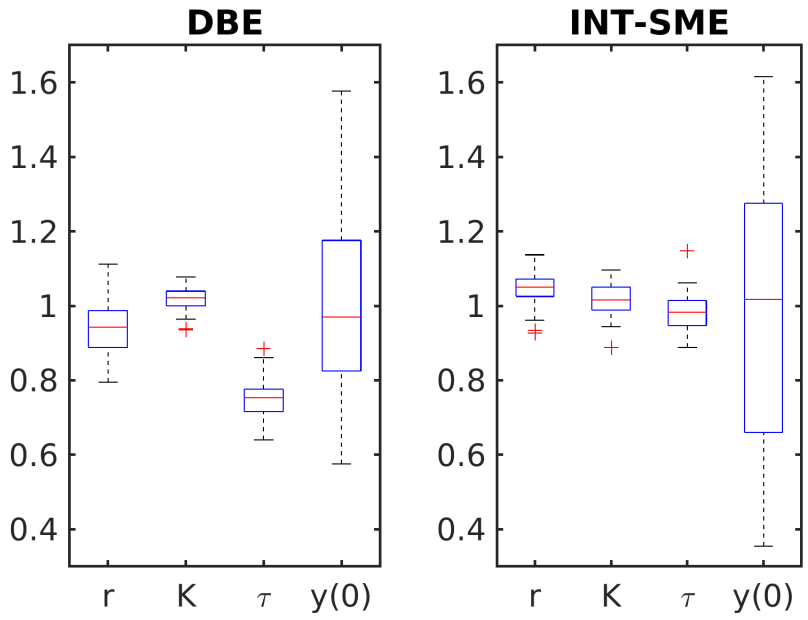


Figure 2.12: Boxplot of p_0's for DBE and INT-SME when applied to Hutchinson's model. Parameters are normalized in this plot. This is for 50 sets of simulated data.

Kermack-McKendrick model

The Kermack-McKendrick model is our second example of a DDE model and it contains two lags. This DDE system models the spread of disease within a population, using a Susceptible-Infected-Recovered (SIR) compartment model. It is a more complicated version of the standard ODE Kermack-McKendrick model [26]. The model is specified

by the system of DDEs,

$$y_1'(t) = -y_1(t)y_2(t-\tau_1) + y_2(t-\tau_2), \tag{2.25}$$

$$y_2'(t) = y_1(t)y_2(t-\tau_1) - y_2(t), \tag{2.26}$$

$$y_3'(t) = y_2(t) - y_2(t-\tau_2), \tag{2.27}$$

where y_1 is the number of susceptible individuals, y_2 is the number of infected individuals, and y_3 is the number of recovered individuals. As in [51], the parameters are chosen to be $\tau_1 = 1$, $\tau_2 = 10$, and $y(t) = y(0) = [5,\ 0.1,\ 1]$, for $t \leq 0$. The true trajectories corresponding to these parameters are shown in Figure 2.13. Both the parameters and initial conditions are estimated in this problem.

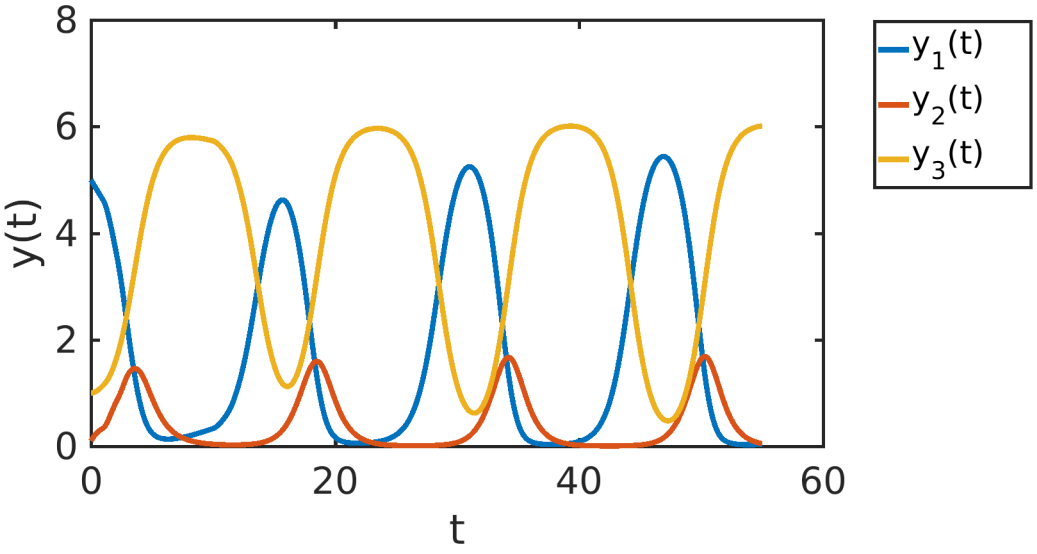


Figure 2.13: True trajectories for the Kermack-McKendrick model

When estimating lag parameters, as we saw in our first example, it is often straightforward to determine reasonable initial bounds on the values they can take. First, since this model is simulated from $t = 0$ to $t = 55$, the lags must lie in this range. Furthermore, the model trajectory is close to being periodic, with a period of around 15 time units. Since the history function is constant, it is reasonable to assume that the lags must lie in $(0, 15)$. Of course, one can further refine this assumption using knowledge of the biological meaning of the lags (τ_1 being the incubation time and τ_2 being the recovery time). We note that if we do not use these refined bounds, then we might encounter a local minimum in the SME objective function (Figure 2.14) at a multiple of τ_2. This local minimum is not present in the INT-SME objective function, but there is a local

minimum at a multiple of τ_1 (Figure 2.14).

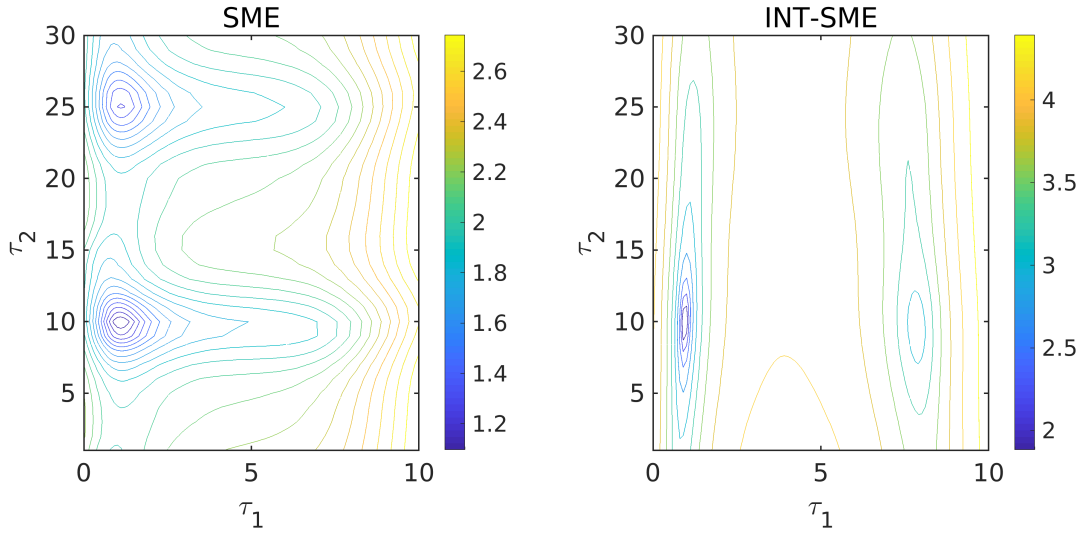


Figure 2.14: Contours of the objective function for SME and INT-SME when applied to the Kermack-McKendrick example. Note, this is with initial conditions fixed at their true values, so only τ_1 and τ_2 are varied. Contours of the objective function for DBE (not shown) are similar to those of SME.

The initial guesses generated by SME, INT-SME, and DBE (with formula (2.19)) are shown in Figure 2.15. All three approaches give p_0's that are sufficiently close to the true parameter values, although DBE results in the least biased initial guesses in this example. For the initial conditions, the guesses are shown in Figure 2.16. Unlike the case of ODEs, DBE and SME could be used to estimate initial conditions for any states whose lagged values appear in f. In this example, only $y_2(0)$ can be estimated in this way. We see that the best initial guesses are obtained by simply using the observation at the initial time. INT-SME's estimate of $y_1(0)$ has slightly less variance, but the estimate is also slightly biased.

2.6.5 Goodwin Example

This system of ODEs models a biological oscillator [19], which has been investigated in applications including enzyme kinetics and circadian clocks [18]. This negative feedback

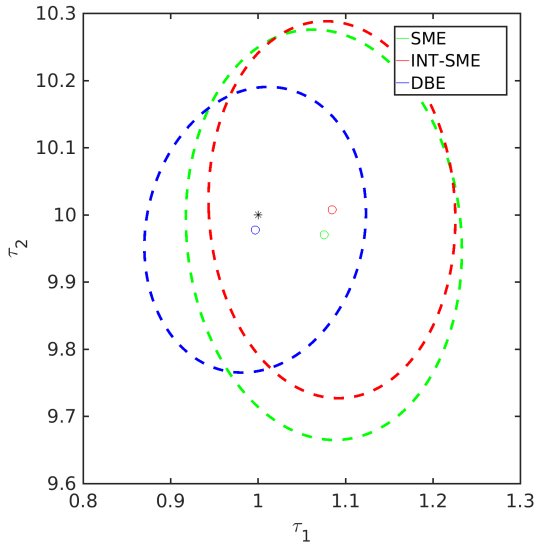


Figure 2.15: 95% confidence ellipse for the p_0's generated by each approach for the Kermack-McKendrick example. Note, this is with initial conditions fixed at their noisy values for SME and DBE, while INT-SME is also estimating the initial conditions.

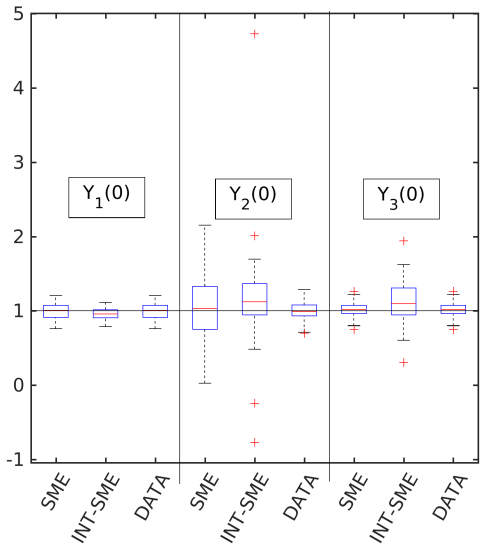


Figure 2.16: Boxplot of the p_0's generated using different approaches for the Kermack-McKendrick example. Since SME can only estimate the initial condition on the second component of the state vector, the estimates for the other two components are based on the noisy data directly.

loop model is given by,

$$y_1'(t) = \frac{a}{A + y_3(t)^\sigma} - by_1(t), \tag{2.28}$$

$$y_2'(t) = \alpha y_1(t) - \beta y_2(t), \tag{2.29}$$

$$y_3'(t) = \gamma y_2(t) - \delta y_3(t), \tag{2.30}$$

where $y_1(t)$ is the product being synthesized, $y_2(t)$ is an intermediate product, and $y_3(t)$ exerts a negative feedback on the synthesis of $y_1(t)$. Initial conditions and other parameters are chosen to be $a = 3.4884$, $A = 2.15$, $b = 0.0969$, $\alpha = 0.0969$, $\beta = 0.0581$, $\gamma = 0.0969$, $\sigma = 10$, $\delta = 0.0775$, $y_1(0) = 0.3617$, $y_2(0) = 0.9137$, and $y_3(0) = 1.3934$, for $t \in [0, 80]$. These values are chosen so that the true trajectories exhibit oscillatory behaviour [2]. The true trajectories corresponding to these parameters are shown in Figure 2.17. Both the parameters and initial conditions are estimated in this problem.

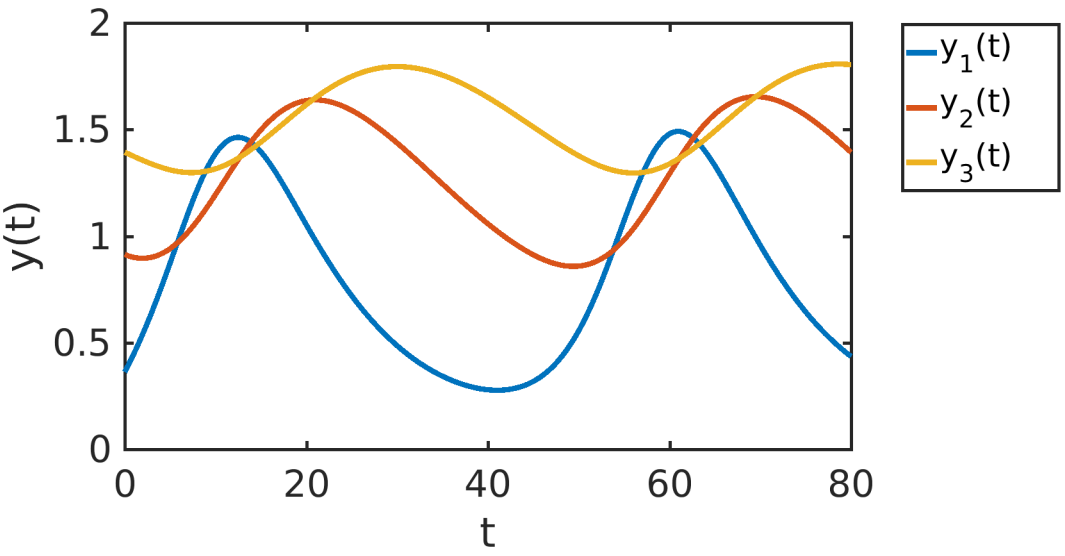


Figure 2.17: True trajectories for the Goodwin model

While it is not the topic of this thesis, we will briefly consider the identifiability of this model. It turns out that, while the parameters are structurally identifiable, they are not practically identifiable. Roughly speaking, this means that some of the parameters may have very large confidence intervals if the data are not sufficiently accurate. We demonstrate this through a simple experiment, where we vary to scale of the noise on a single set of observations. We first accurately simulated the model with the true parameters. We then generated a single set of observations by adding normally distributed noise, with an observation every time unit across the interval. For this set of observations, we found the

best fit parameters and calculated the confidence intervals (using nlparci in MATLAB). We then scaled the noise on the observations and recomputed the best fit parameters and their confidence intervals. This was repeated several times to demonstrate how the level of noise affects the results. This whole procedure is repeated several times and we plot the average values in Figure 2.18. For practically identifiable parameters, we expect the size of the confidence intervals to grow linearly with the level of noise. We see that parameters p_1 and q_1 do not behave this way, as their confidence intervals become much larger as the level of noise increases. With this in mind, we choose a level of noise in our next experiment that is not too large to ensure identifiability (we use noise that is 1% of the average value of the state vector).

The main purpose of this example is to demonstrate how the structure of the model allows us to reduce the size of the problem. We observe that all of the initial procedures result in 3 subproblems, one for each component of the system of ODEs. Furthermore, only one subproblem contains nonlinear parameters. This means that the problem has been reduced from a single optimization over all 11 parameters, to 2 LS problems over 3 parameters each and 1 NLS problem over 2 nonlinear parameters and 3 linear parameters. In our experiment, we have used INT-SME with no smoother. For the nonlinear subproblem, we have used LHS and assumed that the nonlinear parameters are in $[0.01, 100]$. We performed the full estimation procedure on several sets of simulated data. On average, this resulted in the initial guess being generated in 0.076s and the final optimization taking 1.22s. The sensitivities for the final optimization were approximated using forward differences.

2.6.6 Mendes Problem

Our last test problem is a benchmark problem that was originally posed in [38] and has been subsequently studied [44, 43, 2, 17]. In our notation, this model can be expressed as,

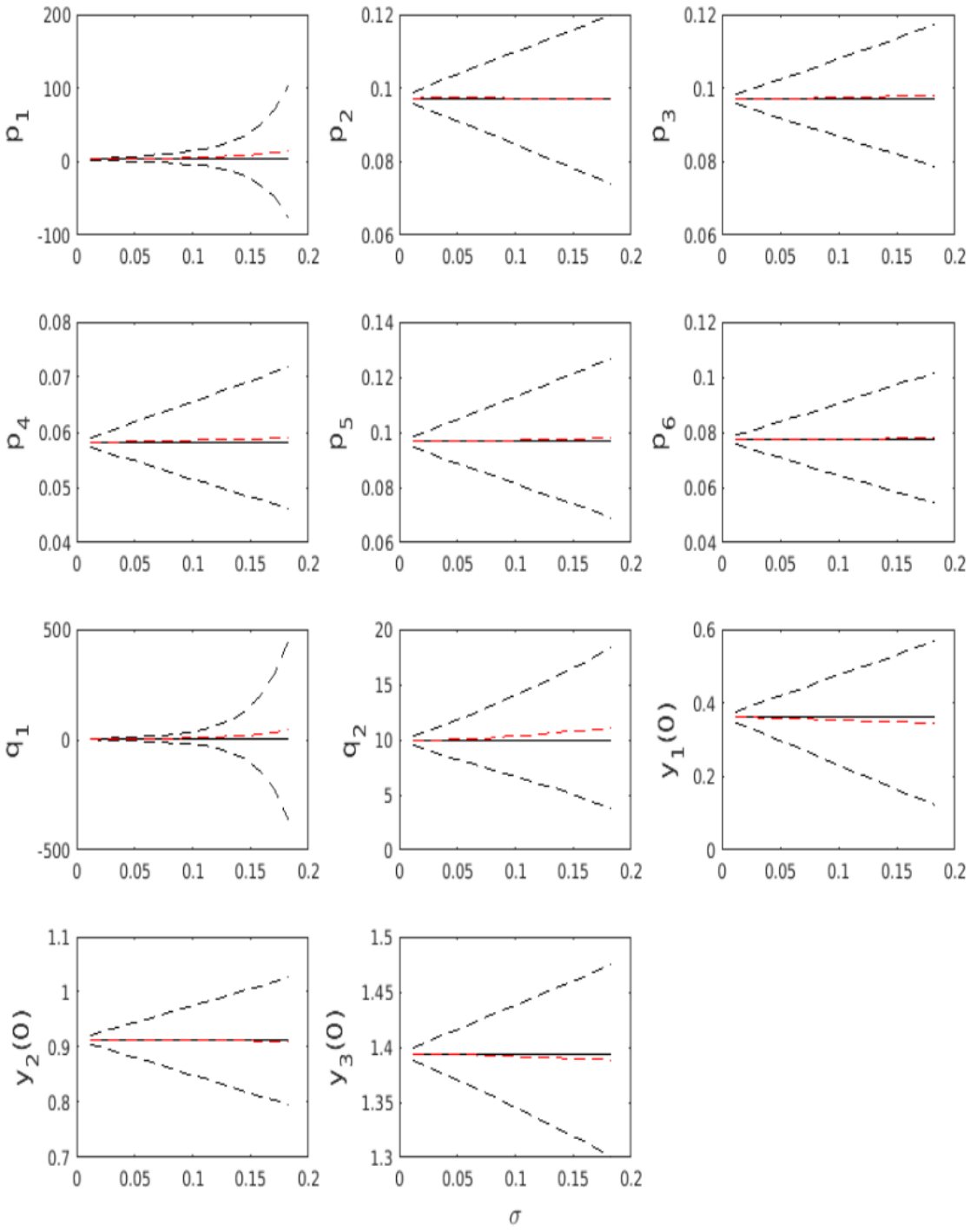


Figure 2.18: Practical identifiability of parameters in the Goodwin model. The x-axis is the level of noise. The dashed black lines are the 95% confidence intervals and, the solid black line is the true value of the parameter, and the red dashed lines are the best fit parameter estimates. All values in the plot are averaged over 50 repetitions of the experiment.

$$
\begin{aligned}
y_1'(t) &= \frac{k_1}{1+\left(\frac{P}{q_1}\right)^{q_2}+\left(\frac{q_3}{S}\right)^{q_4}} - k_2 y_1 \\
y_2'(t) &= \frac{k_3}{1+\left(\frac{P}{q_5}\right)^{q_6}+\left(\frac{q_7}{y_7}\right)^{q_8}} - k_4 y_2 \\
y_3'(t) &= \frac{k_5}{1+\left(\frac{P}{q_9}\right)^{q_{10}}+\left(\frac{q_{11}}{y_8}\right)^{q_{12}}} - k_6 y_3 \\
y_4'(t) &= \frac{k_7 y_1}{y_1+q_{13}} - k_8 y_4 \\
y_5'(t) &= \frac{k_9 y_2}{y_2+q_{14}} - k_{10} y_5 \\
y_6'(t) &= \frac{k_{11} y_3}{y_3+q_{15}} - k_{12} y_6 \\
y_7'(t) &= \frac{k_{13} y_4 (\frac{1}{q_{16}})(S-y_7)}{1+(\frac{S}{q_{16}})+(\frac{y_7}{q_{17}})} - \frac{k_{14} y_5 (\frac{1}{q_{18}})(y_7-y_8)}{1+(\frac{y_7}{q_{18}})+(\frac{y_8}{q_{19}})} \\
y_8'(t) &= \frac{k_{14} y_5 (\frac{1}{q_{18}})(y_7-y_8)}{1+(\frac{y_7}{q_{18}})+(\frac{y_8}{q_{19}})} - \frac{k_{15} y_6 (\frac{1}{q_{20}})(y_8-P)}{1+(\frac{y_8}{q_{20}})+(\frac{P}{q_{21}})}.
\end{aligned}
$$

There are 36 parameters to be estimated in total (k and q). Each parameter is assumed to lie in the range $[10^{-12}, 10^6]$, except for the Hill coefficients $(q_2, q_4, q_6, q_8, q_{10}, q_{12})$, which are assumed to lie in $[0.1, 10]$. The data for this test problem consists of 21 uniformly spaced observations of the 8 state variables, over the interval $[0, 120]$, for each of 16 pairs of values for P and S. The 16 pairs are formed from all combinations of $S \in \{0.1, 0.46416, 2.1544, 10\}$ and $P \in \{0.05, 0.13572, 0.3684, 1\}$. Thus, evaluation of the objective function requires simulating the IVP 16 times, on $[0, 120]$. In total, there are 336 observations. One of these 16 corresponding true trajectories is shown in Figure 2.19. The initial conditions are given by,

$$y(0) = [0.66667, 0.57254, 0.41758, 0.4, 0.36409, 0.29457, 1.419, 0.93464].$$

The initial conditions, as well as S and P, are assumed to be known. The data are generated by simulating the model with the true parameters and adding noise relative to the magnitude of each component of the state vector. The true parameter values are $k_{1-6} = 1$, $k_{7-12} = 0.1$, $k_{13-15} = 1$, $q_{1,3,5,7,9,11,13-21} = 1$, and $q_{2,4,6,8,10,12} = 2$.

To demonstrate the performance of our full estimation procedure (Algorithm 1), we report results for this benchmark problem in Table 2.11. This problem contains 21 nonlinear parameters to be estimated and 15 linear parameters. As we discussed in

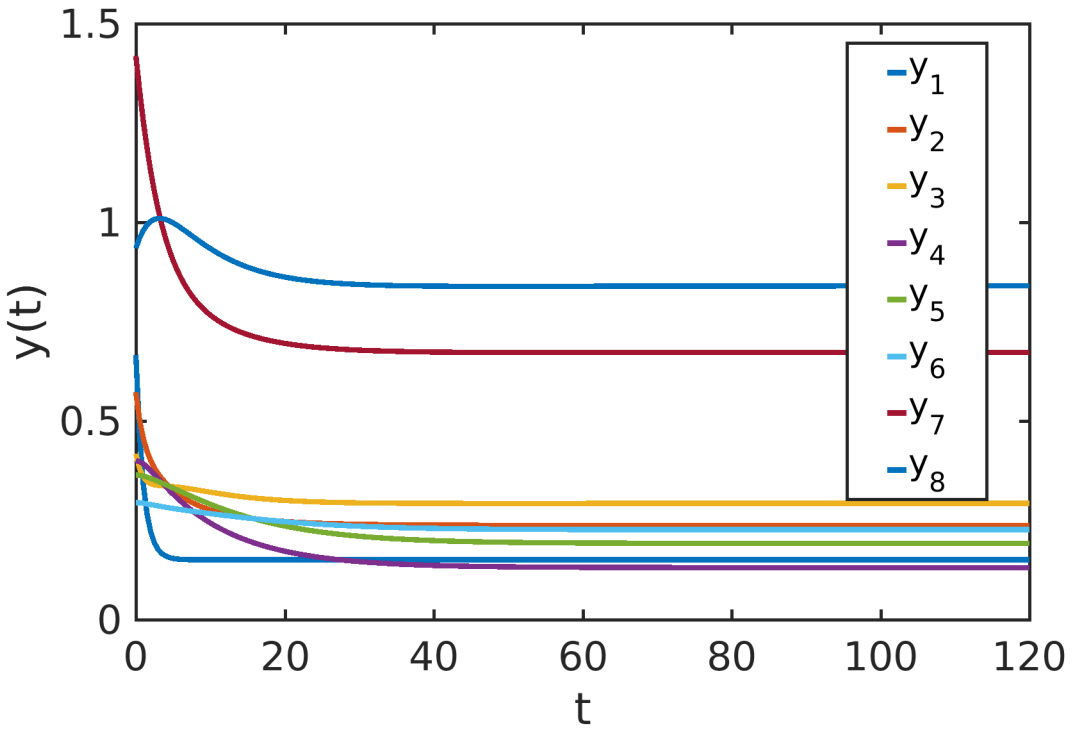


Figure 2.19: True trajectories corresponding to one of the 16 pairs of P and S for the Mendes Problem ($S = 0.46416$ and $P = 1$)

Section 2.3, the structure of the Mendes Problem results in 7 subproblems, as summarized in Table 2.1.

From our numerical results, we see that the choice of smoother and initial procedure can impact the performance. For example, we see that if cubic splines are used and the relative noise is 5%, then we fail to find the best fit parameters half of the time. If INT-SME and a smoother are used, then we succeed more often, although we found using a smooth spline was more successful than lpe in this example.

initial	smoother	Rel Noise (%)	initial time (s)	opt time(s)	total time (s)	success
SME	cubic spline	3	3.51	5.87	9.38	10/10
SME	cubic spline	5	5.10	18.82	23.91	5/10
INT-SME	lpe	5	5.3	16.6	21.9	17/20
INT-SME	splinefit	5	6.5	9.5	16.0	20/20

Table 2.11: Runtimes of our implementation of Algorithm 1 for the Mendes Problem. We also report how many of these runs succeeded. Cubic spline refers to fitting a cubic spline with a knot at each observation time, whereas splinefit uses a coarser mesh to smooth the data. This experiment was run on an Intel X5675 (3.08 GHz).

This experiment was run in Matlab, but the Mex interface was used to allow efficient simulation of the ODE using DDEM, which is written in C++. Mex was also used to make evaluation of f in the initial procedures more efficient. Shortly after first being proposed as a test problem in [38], the best result was around 10^4 seconds using a global-local hybrid optimizer [44]. An improved time of around 300 seconds, using a global scatter search heuristic, was reported later (on a PC Pentium-III/866 MHz) [43]. In a more recent paper, additional heuristics reduce the time required by the scatter search to around 30 seconds [17] (hardware not specified). As we can see, our approach is competitive with this. Note, no parallelism is used in our results, so the timing can be further reduced. For example, the 7 subproblems can be solved in parallel during the first stage and the 16 IVPs can also be simulated in parallel whenever the final objective function is evaluated. As we will see in the next chapter, parallelism can also reduce the cost of performing the final optimization by speeding up the computation of the required sensitivities.

2.6.7 Choice of ODE solver tolerance

The tolerance used to simulate the IVP can impact the cost of the parameter estimation procedure. Enough accuracy should be requested to ensure that the parameter estimates are not being skewed, but if TOL is chosen to be too strict, then extra computer time is being used. In Table 2.12, we report the number of steps taken by DDEM in order to simulate the true trajectory for several of our test problems, for TOL varying from 10^{-6} to 10^{-3}. Assuming TOL $= 10^{-3}$ is sufficiently accurate, we see that using a stricter TOL may result in a modest increase in computer time - ranging anywhere from $1-3$ times the cost in this case. However, we also note that having extra accuracy in the trajectory simulations may lead to fewer iterations for the optimization to converge.

TOL	Barnes	Kermack-McKendrick	FitzHugh	Goodwin	Ca^{2+} Ion	Hutchinson
10^{-6}	50	139	195	45	3727	120
10^{-5}	34	105	137	42	3233	85
10^{-4}	25	79	101	42	2945	63
10^{-3}	17	67	75	42	2792	45

Table 2.12: Steps per simulation with true model parameters for several test problems

γ \ w	0.5	1	2
1	0.26	0.17	0.19
1.1	0.21	0.14	0.15
1.2	0.15	0.10	0.11
1.3	0.11	0.07	0.07
1.4	0.07	0.03	0.04

Table 2.13: Early Termination Results for Barnes Problem. The numbers across the top indicate the value for w and the numbers in the left column indicate the threshold factor, γ. Each entry in the table indicates the fraction of steps saved, for the given threshold factor and value for w.

2.6.8 Early Termination

When evaluating the objective function (1.7), a model trajectory is simulated. Each time the simulation reaches one of the observation points, that observation's contribution to the objective function can be determined. As the value of the objective function accumulates, it may become apparent that the current simulation will result in a large objective function value. Depending on the optimizer being used, one may wish to reduce computation time by terminating such simulations and returning an artificially large objective function value. As an example, we consider the potential gains of using this idea in a global optimizer based on Cross Entropy (CE) [45, 51]. On each iteration of this optimizer, N samples are drawn from the parameter space. The algorithm then computes the objective function for each of the N samples and identifies the N_{elite} 'elite' samples with the best objective function values and uses those samples to update how the samples will be drawn in the next iteration. This means that any sample from the parameter space that won't correspond to an elite can be terminated before its simulation reaches the final observation point. To investigate the potential of using a simple threshold on the objective function value to terminate simulations early, we draw parameters from a uniform distribution, centered at the true value of each parameter. We then perform a series of model simulations, where we vary the width of our uniform distribution. Our uniform distribution is over the range $(p - \frac{p}{w}, p + \frac{p}{w})$, where w determines the width of the distribution. Varying w allows us to simulate the different stages of CE. As the algorithm converges to the true solution, the range it searches in becomes smaller (w becomes larger). We perform 100 model simulations for each parameter range and for each threshold. The threshold is taken to be a multiple of the objective function at the true value of the parameters. The results are given in Table 2.13. In this example, we can save about 20 out of every 100 time steps. However, this requires a good estimate of the expected minimum to be known.

Chapter 3

Approximating Model Sensitivities

Once the first stage of our parameter estimation procedure has generated a suitable p_0, a local optimizer is used to obtain the final parameter estimates. In this chapter, we first motivate our choice to use a Levenberg-Marquardt (LM) optimizer [32], then we investigate how best to approximate the sensitivities (or gradients) required by the optimizer. This is followed by an investigation of how to ensure that we achieve the level of accuracy requested by the user when we use the approximate model sensitivities in the LM optimizer. We conclude the chapter with a brief discussion of how the methods discussed can be applied to the case of DDE IVPs.

3.1 Choice of optimizer

We first recall that the first order variational equations associated with our ODE IVP are defined by taking the partial derivative of both sides of (1.1) with respect to p. This results in,

$$y'_p(t) = f_p + f_y y_p, \tag{3.1}$$

where y_p is the $n_y \times n_p$ matrix, $(y_p)_{ij} = \frac{\partial y_i}{\partial p_j}$. The associated second order variational equations are defined (after taking the partial derivative of both sides of (3.1) with respect to p) by,

$$y'_{pp_k}(t) \;=\; f_{pp_k} + f_y y_{pp_k} + f_{yp_k} y_p + \sum_{l=1}^{n_y} (y_{p_k})_l f_{py_l} + \sum_{l=1}^{n_y} (y_{p_k})_l f_{yy_l} y_p, \tag{3.2}$$

for $k = 1, \ldots, n_p$. Note that y_{pp} is the $n_y \times n_p \times n_p$ tensor, $(y_{pp})_{ijk} = \frac{\partial^2 y_i}{\partial p_j \partial p_k}$. Next, recall

that our scalar objective function is,

$$O(p) \;=\; \sum_{i=1}^{n_o}\sum_{j=1}^{n_y} \frac{\big(\hat{y}_j(t_i) - y_j(t_i,p)\big)^2}{2}.$$

The associated gradient is given by,

$$J(p) = O_p(p) = \sum_{i=1}^{n_o} y_p(t_i,p)^\top \big(y(t_i,p) - \hat{y}(t_i)\big)$$

and the corresponding Hessian is given by,

$$H(p) = O_{pp}(p) = \bar{S}(p) + \hat{S}(p),$$

where,

$$\bar{S}(p) = \sum_{i=1}^{n_o}\sum_{j}^{n_y} \big(y_p(t_i,p)\big)_j^\top \big(y_p(t_i,p)\big)_j,$$

and,

$$\hat{S}(p) = \sum_{i=1}^{n_o}\sum_{j}^{n_y} \big(y_j(t_i,p) - \hat{y}_j(t_i)\big)\big(y_{pp}(t_i,p)\big)_j.$$

The Gauss-Newton (GN) and LM algorithms both assume that $\bar{S}$ dominates $\hat{S}$. This leads to the approximation, $H \approx \bar{S}$. Assuming we have found an appropriate p_0 to start our minimization, this should be a valid assumption and we expect to observe super linear convergence without having to simulate the solution of the second order variational equations. We note that the original system of ODEs consists of a state vector of length n_y, the first order variational equations (3.1) have $n_y n_p$ additional components, and the second order variational equations (3.2) have another additional $n_y n_p n_p$ components.

3.1.1 BFGS Optimizer

Another commonly used quasi-Newton algorithm is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. This algorithm approximates the Hessian on each iteration by performing two rank-one updates to the approximate Hessian from the previous iteration. The initial Hessian is often chosen to be the identity matrix, but any approximation can be used. Unlike LM and GN, BFGS only requires $J(p)$, not $\bar{S}(p)$. As such, we don't need

to explicitly approximate each $y_p(t_i, p)$, but can apply the adjoint method to directly approximate $J(p)$. We discuss the adjoint method in Chapter 4.

The BFGS approximate Hessian, B, is updated from iteration k to $k+1$ by the formula,

$$B_{k+1} = B_k + \frac{v_k v_k^\top}{v_k^\top s_k} - \frac{B_k s_k s_k^\top B_k^\top}{s_k^\top B_k s_k},$$

where,

$$v_k = J(p_{k+1}) - J(p_k) \text{ and } \; s_k = p_{k+1} - p_k.$$

3.1.2 Numerical Experiment

To determine which optimization algorithm we should use, we investigate their performances when applied to two of our test problems - the Barnes problem (Section 2.6.2) and the FitzHugh-Nagumo model (Section 2.6.1) - in Matlab. For the purposes of this experiment, we treat the initial conditions as fixed for each of these problems, so each problem has 3 parameters to be estimated. These two test problems have $n_y = 2$.

We have generated 100 sets of noisy data for both test problems and performed the least squares minimization from a different random initial p_0 for each data set. Each p_0 is chosen to be close to the optimal p, such that the optimizers have little difficulty converging. We report the average time each optimizer takes to converge in Table 3.1. The 3 optimizers we use are BFGS, Gauss-Newton (GN), and Newton's method (NM). Note, we have used the variational approach to obtain the gradient for BFGS, although the adjoint approach could be used to possibly improve its performance. For BFGS, we try two different choices for the initial Hessian approximation. For the Barnes problem, we try both the identity matrix and the Hessian approximation used by GN. For the FitzHugh-Nagumo model, we only use the GN Hessian approximation.

model	BFGS (H_0 = identity)	BFGS ($H_0 = \bar{S}(p)$)	NM	GN
Barnes	0.447s	0.132s	0.238s	0.0887s
FitzHugh	–	0.530s	1.51s	0.433s

Table 3.1: Cost in seconds of BFGS, GN, and NM for the Barnes and FitzHugh test problems.

We see that GN is the most efficient optimizer, at least for these test problems. BFGS is about 5 times slower and NM is about 2.5 times slower. If we use BFGS with the initial Hessian chosen to be the Hessian used by GN, then we achieve performance closer to that

of GN - which suggests it might be using the adjoint method to compute the gradient more efficiently. These results also indicate that the GN Hessian approximation is quite good. We also tried using the full Hessian to initialize BFGS, but found it to be less effective than using the GN Hessian approximation. Note, LM should perform about the same as GN - the difference being that LM will be more robust in cases where our initial p_0 is not sufficiently close to the optimal p. In such cases, it is likely that BFGS will not be as effective as it is here since the initial GN Hessian approximation will be less justified.

Also, we looked at the time it takes to simulate the model, the first order variational equations, and the second order variational equations. For the Barnes problem, the times were 0.0039s, 0.014s, and 0.037s, respectively. For the FitzHugh-Nagumo model, the times were 0.0097s, 0.041s, and 0.182s, respectively. The differences in cost is roughly consistent with the fact that the systems of ODEs contain 2 components, 8 components, and 26 components, respectively. We note that the second order variational equations could be more efficiently implemented, due to the equality of mixed partials. That is, we could reduce the number of components from 26 down to 20. Or more generally, reduce it from $n_y + n_y n_p + n_y n_p n_p$ to $n_y + n_y n_p + n_y \frac{n_p(n_p+1)}{2}$.

Based on these results, using a GN iteration is likely to be more efficient for our application than a full Newton iteration, due to the high cost of simulating the second order variational equations. If the BFGS algorithm is to be used, it is advisable to choose the initial Hessian approximation to be the GN approximation. For some problems, it is well known that the gradient can be more efficiently computed using the adjoint method than using the variational approach we use here. The adjoint method for computing the gradient is described in Chapter 4. In any case, we will want to be able to efficiently approximate the model sensitivities in order to use the GN approximation of the Hessian.

We now review several approaches for numerically approximating the model sensitivities for ODE IVPs - finite differences and two different ways to simulate the variational equations. To demonstrate each of the methods for approximating these sensitivities, we consider the Calcium Ion test problem (Section 2.6.3) and the Barnes problem (Section 2.6.2). We also discuss how each approach can exploit parallelism. All experiments in the next sections are performed on a machine with two Intel E5-2697v2 12-core processors. The computationally expensive parts of the experiments are performed in C++, with the Mex interface used to communicate the results back to Matlab, where all timing and visualization of the results is performed.

3.2 Finite Differences

To approximate $y_{p_k}(t)$, the standard forward difference (FD) approximation is,

$$\frac{y(t, p + \epsilon_{FD} e_k) - y(t, p)}{\epsilon_{FD}} \approx y_{p_k}(t) + O(\epsilon_{FD}), \tag{3.3}$$

and the centered difference (CD) approximation is,

$$\frac{y(t, p + \epsilon_{CD} e_k) - y(t, p - \epsilon_{CD} e_j)}{2\epsilon_{CD}} \approx y_{p_k}(t) + O(\epsilon_{CD}^2). \tag{3.4}$$

The usual recommended values are $\epsilon_{FD} = \sqrt{\epsilon_{mach}}$ and $\epsilon_{CD} = \epsilon_{mach}^{\frac{1}{3}}$. However, this assumes that we have approximated $y(t)$ to machine precision. If we have only approximated $y(t)$ with a tolerance of ϵ_{ODE}, then these choices are not necessarily optimal, as they can amplify the approximation error in the difference between the two simulated trajectories. By choosing a larger perturbation, we increase the truncation error of the finite difference approximation of the gradient, but we reduce the factor by which the simulation approximation error is amplified. With this in mind, using $\epsilon_{FD} = \sqrt{\epsilon_{ODE}}$ and $\epsilon_{CD} = \epsilon_{ODE}^{\frac{1}{3}}$ is expected to balance the impact of these errors.

Numerical results for the Barnes problem are shown in Tables 3.2 and 3.3. We see that these choices for ϵ_{FD} and ϵ_{CD} are too conservative for this example. This means that the approximation errors in $y(t, p + \epsilon_{FD} e_k)$ and $y(t, p)$ are cancelling out and not getting amplified by the $\frac{1}{\epsilon_{FD}}$ factor. This is to be expected if the simulations are taking similar step sizes, since they are using the same underlying CRK formula. For this example, we also note that when $\epsilon_{FD} = \epsilon_{mach}^{\frac{1}{2}}$, the cancellation error is small even at relaxed tolerances. By choosing $\epsilon_{FD} = \epsilon_{ODE}$, we obtain similar accuracy. Using $\epsilon_{FD} = \sqrt{\epsilon_{ODE}}$, we obtain less accuracy due to overcompensating for the potential approximation error by increasing the truncation error.

We also repeat this for the Calcium Ion test problem, with results shown in Tables 3.4 and 3.5. For FD, when $\epsilon_{FD} = \epsilon_{mach}^{\frac{1}{2}}$, the cancellation error dominates at relaxed tolerances. By using $\epsilon_{FD} = \epsilon_{ODE}$, we significantly reduce the error at relaxed tolerances. Unlike the Barnes problem, the Calcium Ion model is slightly stiff, so the step sizes used by the ODE solver are not as similar between each perturbed trajectory. This means that the approximation error can dominate if ϵ_{FD} is chosen too small relative to ϵ_{ODE}. At relaxed tolerances, $\epsilon_{CD} = \sqrt{\epsilon_{ODE}}$ results in the truncation error of CD dominating the calculation. This can be improved by instead choosing a slightly smaller perturbation, $\epsilon_{CD} = \sqrt{\frac{\epsilon_{ODE}}{10}}$.

ϵ_{ODE}	time (s)	max error in approximation to $y_p(t_i)$		
		$\epsilon_{FD} = \sqrt{\epsilon_{ODE}}$	$\epsilon_{FD} = \epsilon_{ODE}$	$\epsilon_{FD} = \sqrt{\epsilon_{mach}}$
1e-03	0.0016	5.89e-01	1.89e-02	2.82e-02
1e-04	0.0017	2.06e-01	3.84e-03	4.68e-03
1e-05	0.0018	6.72e-02	1.14e-03	1.03e-03
1e-06	0.0022	2.14e-02	4.13e-05	3.44e-05
1e-07	0.0022	6.76e-03	5.02e-06	2.51e-05
1e-08	0.0025	2.14e-03	1.36e-05	1.01e-05
1e-09	0.0030	6.77e-04	1.81e-04	1.13e-05

Table 3.2: Errors of forward differences for varying simulation tolerance, ϵ_{ODE}, and for three different perturbations, for the Barnes test problem.

ϵ_{ODE}	time (s)	max error in approximation to $y_p(t_i)$		
		$\epsilon_{CD} = \epsilon_{ODE}^{\frac{1}{3}}$	$\epsilon_{CD} = \sqrt{\epsilon_{ODE}}$	$\epsilon_{CD} = \epsilon_{mach}^{\frac{1}{3}}$
1e-03	0.0029	9.41e-01	9.19e-02	2.82e-02
1e-04	0.0031	2.22e-01	1.14e-02	4.67e-03
1e-05	0.0033	4.84e-02	8.53e-04	1.22e-03
1e-06	0.0036	1.05e-02	9.79e-05	3.32e-05
1e-07	0.0040	2.26e-03	1.04e-05	3.28e-06
1e-08	0.0045	4.88e-04	5.71e-07	7.31e-07
1e-09	0.0055	1.05e-04	1.04e-07	3.94e-08

Table 3.3: Errors of central differences for varying simulation tolerance, ϵ_{ODE}, and for three different perturbations, for the Barnes test problem.

ϵ_{ODE}	time (s)	max error in approximation to $y_p(t_i)$	
		$\epsilon_{FD} = \epsilon_{ODE}$	$\epsilon_{FD} = \sqrt{\epsilon_{mach}}$
1e-03	0.035	1.54e+00	1.28e+05
1e-04	0.039	3.15e-02	2.75e+01
1e-05	0.043	1.02e-02	7.08e+00
1e-06	0.051	1.31e-03	7.02e-02
1e-07	0.060	1.02e-04	5.03e-04
1e-08	0.076	7.02e-04	5.90e-04
1e-09	0.103	1.07e-03	9.72e-05

Table 3.4: Errors of forward differences for varying simulation tolerance, ϵ_{ODE}, and for two different perturbations, for the Calcium Ion test problem.

3.2.1 Parallel Finite Differences

Assuming that we divide up the work between N_p processors (as evenly as possible), then the maximum number of trajectory calculations that at least one of the processors will have to perform will be $\lceil \frac{n_p+1}{N_p} \rceil$ for FD and $\lceil \frac{2n_p}{N_p} \rceil$ for CD. Experimental results are shown in Figure 3.1 and Figure 3.2, along with the theoretical speedups.

ϵ_{ODE}	time (s)	max error in approximation to $y_p(t_i)$		
		$\epsilon_{CD} = \sqrt{\epsilon_{ODE}}$	$\epsilon_{CD} = \sqrt{\frac{\epsilon_{ODE}}{10}}$	$\epsilon_{CD} = \epsilon_{mach}^{\frac{1}{3}}$
1e-03	0.065	1.36e+00	**1.36e-01**	1.93e+02
1e-04	0.070	1.34e-01	**1.37e-02**	2.93e-01
1e-05	0.076	1.33e-02	**1.66e-03**	1.20e-02
1e-06	0.087	1.34e-03	2.14e-04	1.05e-04
1e-07	0.104	1.30e-04	9.51e-05	1.11e-05
1e-08	0.131	1.32e-05	1.21e-06	7.06e-07
1e-09	0.172	2.31e-06	4.08e-07	4.13e-07

Table 3.5: Errors of central differences for varying simulation tolerance, ϵ_{ODE}, and for three different perturbations, for the Calcium Ion test problem.

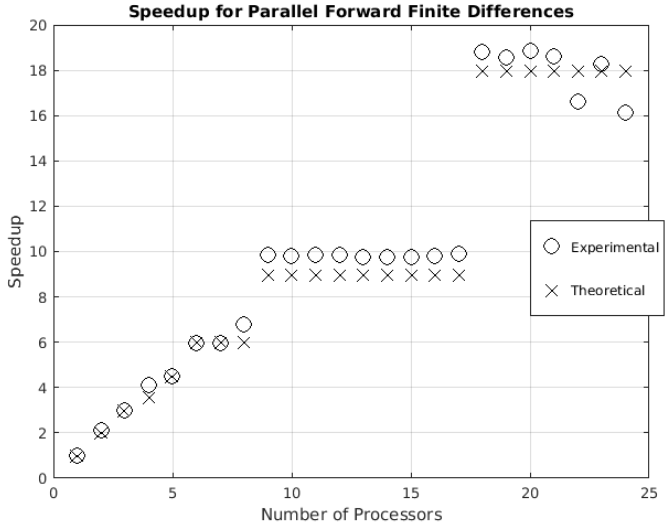


Figure 3.1: Experimental results demonstrating how our parallel version of FD scales with the number of processors. This is for the Calcium Ion test problem.

We see that there is good agreement between our experimental speedups and the theoretical speedups. When N_p is small relative to the number of trajectories we need to compute, we see that we get close to optimal speedups. As expected, using finite difference approximations in parallel results in near optimal speedups whenever the number of parameters is evenly divisible by the number of processors.

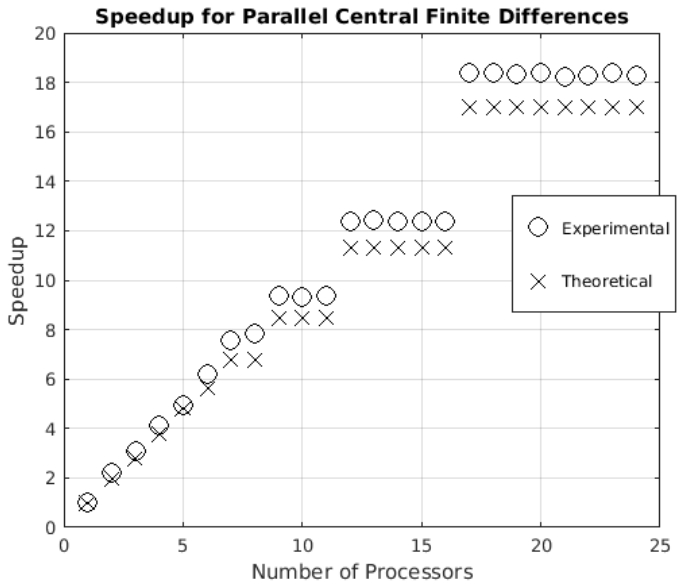


Figure 3.2: Experimental results demonstrating how our parallel version of CD scales with the number of processors. This is for the Calcium Ion test problem.

3.3 Variational Approach

As we discussed, the first order variational equations are given by (3.1). To fully specify the variational IVP, we have that the initial condition for each $\frac{\partial y_i}{\partial p_j}$ is,

$$\frac{\partial y_i}{\partial p_j}(0) = \begin{cases} 1, & \text{if } p_j \text{ is the initial condition for } y_i \\ 0, & \text{otherwise.} \end{cases}$$

This resulting matrix valued IVP can be approximated simultaneously with the original system (1.1). A potential disadvantage of this approach is that the variational system consists of $n_y + n_y n_p$ differential equations. Numerical results for the variational approach applied to the Calcium Ion test problem and the Barnes problem are shown in Table 3.6. We see that the variational approach is quite effective for both of these examples. The error in the approximation of $y_p(t)$ is consistent with the specified user tolerance (except at the most relaxed tolerance for the Calcium Ion problem) and the cost is not drastically increasing as more accuracy is requested.

	Calcium Ion		Barnes	
TOL	time (s)	max error in approximation of $y_p(t_i)$	time (s)	max error in approximation of $y_p(t_i)$
1e-03	0.036	4.21e-02	0.0009	9.67e-04
1e-04	0.032	8.96e-05	0.0010	9.41e-05
1e-05	0.038	1.30e-05	0.0012	1.10e-05
1e-06	0.046	1.96e-06	0.0014	1.28e-06
1e-07	0.060	1.23e-07	0.0016	1.67e-07
1e-08	0.078	1.40e-08	0.0020	1.87e-08
1e-09	0.107	1.40e-09	0.0025	1.69e-09

Table 3.6: For varying simulation tolerance, we report the associated max error in the approximation to $y_p(t_i)$ for the Calcium Ion and Barnes test problems. (We simulate the variational equations using DDEM - with the standard defect control (INT=0) and force the solver to hit each observation time.)

3.3.1 Parallel Variational Approach

The maximum number of parameters that a single processor will be responsible for will be $\lceil \frac{n_p}{N_p} \rceil$, where N_p is the number of processors. Note that unlike the divided differences case, where each processor is simulating the same system of ODEs, each processor is now simulating a subset of the full variational system of ODEs. Depending on how the parameters appear in f, it is possible that some processors will have systems of ODEs that require more work to compute than the other processors. Also, assuming we are doing error control on $y_p(t)$, we will obtain different numerical approximations if we vary N_p. An example of this for the Calcium Ion test problem is shown in Table 3.7.

Experimental results are shown in Figure 3.3, along with the theoretical speedups based on fitting the cost model,

$$\text{cost} = c_0 + c_1 \lceil \frac{n_p}{N_p} \rceil,$$

to the experimental data. We see that there is fairly good agreement between our experimental speedups and the theoretical model of speedups. I am not sure why we observe the jumps in speedup between odd and even numbers of processors, but suspect it has to do with the hardware, since it only occurs after we get above 12 processors. We note that we don't get as much speedup with the variational equations compared to finite

# of Processors	max error in approximation of $y_p(t_i)$
1	8.96e-05
2	5.99e-05
3-4	2.78e-04
5	2.76e-04
6-8	3.72e-04
9-16	3.82e-04
17	3.73e-04

Table 3.7: Experimental results demonstrating how a parallel version of the Variational approach results in different numerical results as we vary N_p. This is for the Calcium Ion test problem, with a tolerance of 10^{-4}.

differences, since the amount of work for each parameter is much smaller for the variational equations. This can be seen by considering the fact that our model parameters came out to be $c_0 = 0.06s$ and $c_1 = 0.01s$. Roughly speaking, this means that the cost of simulating $y(t)$ and one column of $y_p(t)$ is more expensive than each additional column of $y_p(t)$. This makes sense, since we require the evaluation of $f_y(t)$, regardless of how many columns of $y_p(t)$ we are approximating. For the Calcium Ion test problem, $f_y(t)$ is fairly expensive to evaluate due to the functional form of f.

We can also verify that $c_0 = 0.06s$ and $c_1 = 0.01s$ make sense for the Calcium Ion test problem by counting the operation counts for f, f_y, f_{p_k}, $f_y y_{p_k}$, and $f_y y_p + f_{p_k}$, for an arbitrary p_k. These turn out to be roughly 50 flops, 100 flops, 4 flops, 16 flops, and 4 flops, respectively. This means that the right hand side function for the Variational approach requires 150 flops no matter how many parameters are being considered and the cost associated with a single parameter is roughly 24 flops. This ratio of $\frac{150}{24} \approx 6$ is consistent with the experimental ratio of $\frac{c_0}{c_1} \approx 6$. Breaking down the cost in this way, it is also easy to determine what the best speedup is expected to be. In this case, we expect the maximum speedup to be roughly 3.3, which is again in agreement with our numerical results.

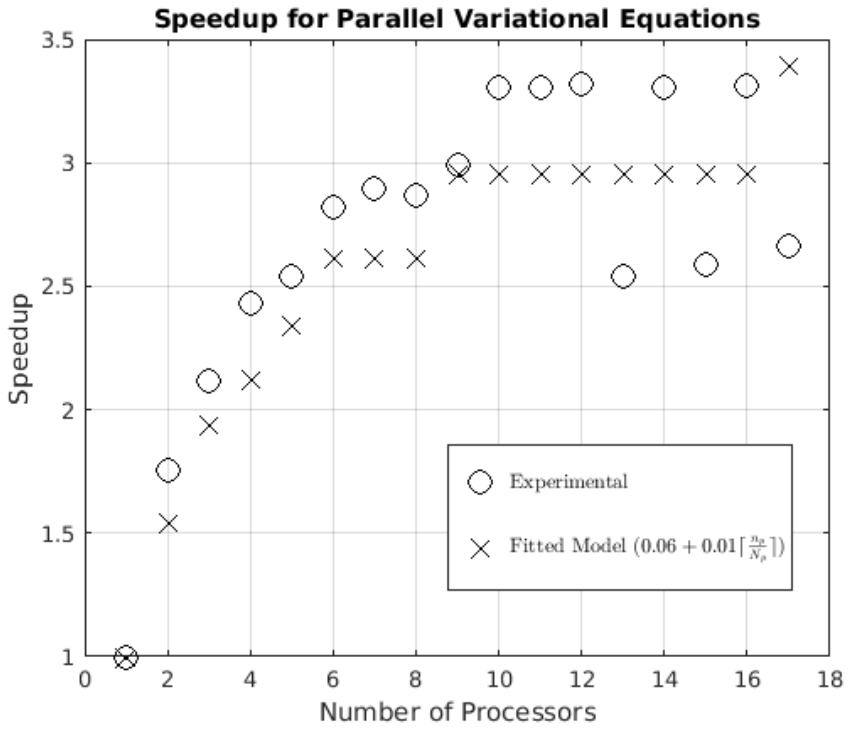


Figure 3.3: Experimental results demonstrating how our parallel version of the Variational approach scales with the number of processors. This is done for the Calcium Ion test problem.

3.4 Green's Function Method (GFM)

It has been suggested that the variational equations can sometimes be simulated more efficiently by making use of a Green's function method (GFM) [25, 28]. The motivation is that GFM reduces the variational equations to $n_y + n_y^2$ differential equations and $n_y n_p$ integrals.

The Green's function kernel is denoted by, $K(t, \tau)$, and can be shown to satisfy,

$$\frac{dK(t,\tau)}{dt} = \frac{\partial f}{\partial y}(t)K(t,\tau),\ K(\tau,\tau) = I_{n_y}, \tag{3.5}$$

where I_{n_y} is the $n_y \times n_y$ identity matrix.The sensitivities can then be expressed in integral form as,

$$\frac{\partial y}{\partial p}(t) = K(t,0)\frac{\partial y}{\partial p}(0) + \int_0^t K(t,\tau)\frac{\partial f}{\partial p}(\tau)\,d\tau. \tag{3.6}$$

What we notice with this integral formulation is that we have the difficulty that we require both $K(t, 0)$ (where we need the first argument to vary) and $K(t, \tau)$ (where, for

a fixed t, we need the second argument to vary). This has led to two different ways to use (3.6) - the forward approach and the backward (adjoint) approach. The adjoint method discussed in the next chapter is similar to the backward GFM approach, so we only consider forward GFM in this chapter.

3.4.1 Forward GFM

Using the fact that $K(r,t) = K(r,s)K(s,t)$, it follows that (3.6) can be rewritten as,

$$y_p(t+\Delta t) = K(t+\Delta t, t)y_p(t) + \int_t^{t+\Delta t} K(t+\Delta t, \tau) f_p(\tau)\, d\tau. \tag{3.7}$$

This gives us an explicit formula to step $y_p(t)$ forward in time, assuming we have an approximation of $y(t)$ available. Since we are stepping in time using this recurrence, it is necessary to approximate the $K(t+\Delta t, t)$ on each step with sufficient accuracy to ensure that the error in each $y_p(t_i)$ is still bounded by a small multiple of the user specified tolerance. Furthermore, we still have the difficulty that we need $K(t+\Delta t, \tau)$ in the integral term, which essentially means we need to be simulating (3.5) for a range of values of τ when approximating the integral. We now present a standard technique for approximating the solution of (3.5) using the Magnus expansion.

3.4.2 Piecewise Magnus method

The Magnus solution of (3.5) is given by $K(t+\Delta t, t) = \exp(\Omega(t+\Delta t, t))$, where $\Omega(t+\Delta t, t)$ is the Magnus series [35]. The piecewise Magnus method (PMM) again makes use of the fact that $K(r,t) = K(r,s)K(s,t)$. This means that if we want $K(t+\Delta t, t)$, but our numerical approximation of $\Omega(t+\Delta t, t)$ is too poor, we can simply compute the Magnus solution at intermediate values of t and do a sequence of matrix multiplications to obtain a more accurate approximation to $\Omega(t+\Delta t, t)$ [12]. To approximate the Magnus series solution of (3.5) in our implementation, we use the fourth order numerical approximation schemes [8] for $\Omega(t,\tau)$,

$$\Omega(t+\Delta t, t) = \frac{\Delta t}{6}\big(f_y(t) + 4f_y(t+\frac{\Delta t}{2}) + f_y(t+\Delta t)\big) - \frac{\Delta t^2}{12}[f_y(t), f_y(t+\Delta t)] \tag{3.8}$$

and,

$$\Omega(t+\Delta t, t) = \frac{\Delta t}{2}\big(f_y(t_a) + f_y(t_b)\big) - \frac{\sqrt{3}\Delta t^2}{12}[f_y(t_a), f_y(t_b)], \tag{3.9}$$

where $t_a = t + \frac{\Delta t}{2} + \frac{\sqrt{3}}{6}\Delta t$, $t_b = t + \frac{\Delta t}{2} - \frac{\sqrt{3}}{6}\Delta t$, and $[A, B]$ denotes the commutator of A and B, $AB - BA$. To do step size selection and error control on $K(t + \Delta t, t)$, we use the norm of the difference between the two above approximations as our error criterion.
Once we have found a Δt such that the error estimate associated with $K(t+\Delta t, t)$ is small enough, we then approximate the integral, $\int_t^{t+\Delta t} K(t + \Delta t, \tau) f_p(\tau) d\tau$. We use the Gauss-Lobatto quadrature rule with 4 points to approximate this integral. We chose this formula since it allows endpoint values to be reused across time steps and in our experiments it was found to give a good balance between accuracy and efficiency compared to the 3 point and 5 point quadrature rules. To compute the integrands at the interior quadrature points, we use (3.8). Numerical results for the Calcium Ion test problem and Barnes problem are shown in Table 3.8. For these results, we have chosen a tolerance on $K(t + \delta t, t)$, such that we achieve the expected accuracy in $y_p(t)$. Experimentally, we found that the tolerance on $K(t + \delta t, t)$ should be roughly equal to the tolerance on $y(t)$ divided by the number of PMM steps. Since we are using a fourth order Magnus approximation, we see that the method becomes inefficient when high accuracy is required, especially for the Calcium Ion test problem.

TOL on $y(t)$	TOL on $K(t + \delta t, t)$	time (s)	max error in $y_p(t_i)$	PMM steps
1e-03	5e-07	0.042	2.364025e-03	1749
1e-05	5e-09	0.052	2.391480e-05	2622
1e-07	5e-14	0.364	3.908701e-07	21060
1e-09	5e-16	0.646	4.183157e-09	40377
1e-03	1e-04	0.000486	2.256896e-03	52
1e-05	1e-06	0.000734	4.876296e-05	98
1e-07	1e-09	0.002157	9.019444e-07	369
1e-09	1e-11	0.004830	1.114407e-08	906

Table 3.8: Performance of our implementation of the forward Green's function method using PMM. These results are for the Calcium Ion test problem (top) and the Barnes problem (bottom). Both problems have $n_o = 11$.

3.4.3 Parallel forward GFM

For the forward GFM, it turns out that there is very little real benefit to parallelism over the parameters. We verified this experimentally, but it is easily seen by the fact that the majority of the computation done in GFM involves the Green's function kernel, which is independent of the parameters. Roughly, only about 5% of the cost can be attributed to

the parameters for our Calcium Ion test problem, so it is not surprising that we achieve practically no speedup when assigning the parameters across processors.

However, since the GFM formulation takes a form like, $y_p(t + \Delta t) = A(t)y_p(t) + B(t)$, it is possible to distribute time intervals across processors. That is, each processor would be responsible for approximating $A(t)$ and $B(t)$ over their specified interval(s). The processor with the first time interval would advance the solution of $y_p(t)$ to the end of its time interval, then pass it to the next processor, which will already have its approximations to $A(t)$ and $B(t)$, so it just has to use them to advance $y_p(t)$ through the interval and then pass it to the next processor. Note, this assumes that we aren't controlling the error on $y_p(t)$ directly, but rather the error associated with $A(t)$.

In our implementation, we define each time interval to be one step taken by the CRK solver used to simulate $y(t)$. Roughly speaking, from our experimental results for the Calcium Ion test problem, the simulation of $y(t)$ takes about 6% of the computer time, computing the Green's function kernel takes about 88%, and the propagation of $y_p(t)$ takes about 6%. The expected (theoretical) speedup, $S_{\text{theo}}(N_p)$, can be calculated as,

$$S_{\text{theo}}(N_p) = \frac{1}{f_{seq} + \frac{1-f_{seq}}{N_p}},$$

wheref_{seq} is the fraction of computation that is not parallelizable and N_p is the number of processors.

As an example, we have calculated the sensitivities for the Calcium Ion test problem in parallel. The results are shown in Figure 3.4, along with the expected speedups based on how much of the calculation has to be done sequentially (i.e. about 6%). The experiments are performed on a machine with two 12-core processors. We observe good agreement between the expected speedup and our experimental results. We repeated this for the Barnes problem and obtained similar performance (results not shown). The speedups are less substantial in this case, as f_{seq} is around 30%, since approximating the Green's function kernel is less expensive for the Barnes problem than for the Calcium Ion test problem. For both examples, we chose the tolerances so that the error in the sensitivities was around 10^{-4}.

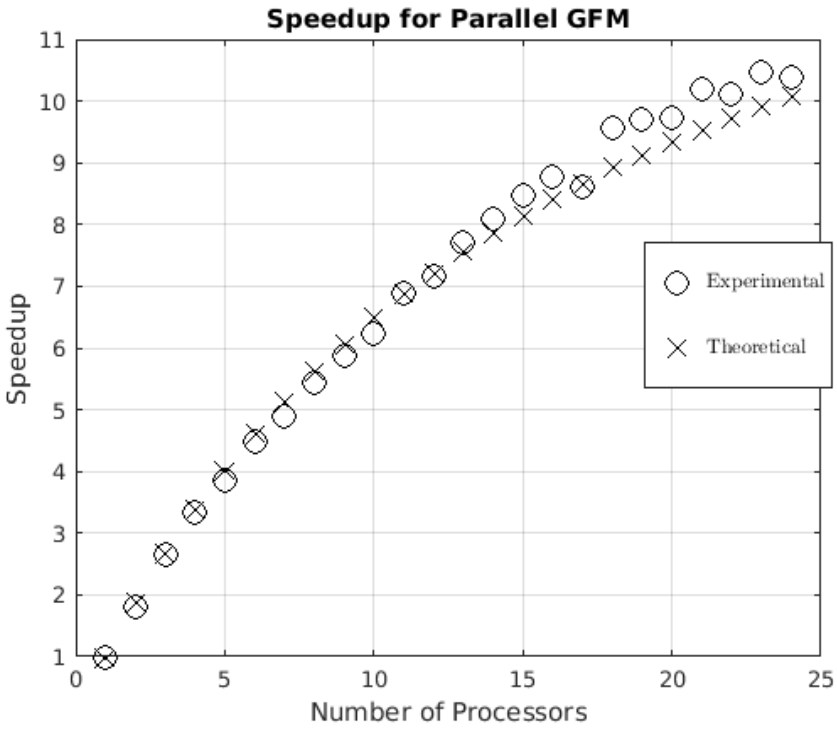


Figure 3.4: Experimental results demonstrating how our parallel version of the GFM approach scales with the number of processors. This is done for the Calcium Ion test problem.

3.5 Comparing Approaches for approximating $y_p(t)$

We summarize the most notable costs of each approach in Table 3.9. The divided difference approaches are straightforward to implement and do not require any extra effort to be taken by the user to provide f_y and f_p, but they do require $y(t)$ to be approximated with more accuracy. The main difference between simulating the variational equations and GFM is that the variational equations consist of a system of $n_y+n_yn_p$ ODEs, whereas GFM consists of a system of n_yn_y ODES and n_yn_p integrals. Since quadratures can be more efficiently approximated than ODEs, GFM can be more efficient in some situations. Roughly speaking, if n_p is large compared to n_y, we expect GFM to perform better than direct simulation of the variational equations.

3.5.1 Comparing Parallel Versions

We also consider how the parallel versions of these methods compare. For each approach, we have plotted the expected and actual computer time versus the number of processors in Figure 3.5 for the Calcium Ion test problem. For each method, we have chosen

method	TOL on $y(t)$	remarks
FD	strictest	$n_p + 1$ trajectories most limited accuracy
CD	stricter	$2n_p$ trajectories limited accuracy
Vari	normal	requires f_y and f_p direct error control $n_y + n_y n_p$ ODEs
GFM	strict	requires f_y and f_p indirect error control $n_y n_y$ ODEs and $n_y n_p$ integrals

Table 3.9: Comparison of Approaches for approximating model sensitivities

appropriate tolerances such that the max error in $y_p(t)$ is around 10^{-4}. The expected computer times are based on the theoretical speedups discussed in the previous sections.

We see that in a sequential environment, the variational approach is significantly faster than the other approaches for this test problem. However, once we have sufficiently many processors, all approaches take roughly the same amount of time. If we had 34 available processors, we should find that CD would become cheaper than FD (assuming the additional cost of that many threads doesn't begin to dominate), since at this point each processor would only have one trajectory to compute and CD uses a more relaxed solver tolerance than FD. For this test problem, hardware, and tolerance, FD with 18 processors turns out to be slightly faster than the other three approaches.

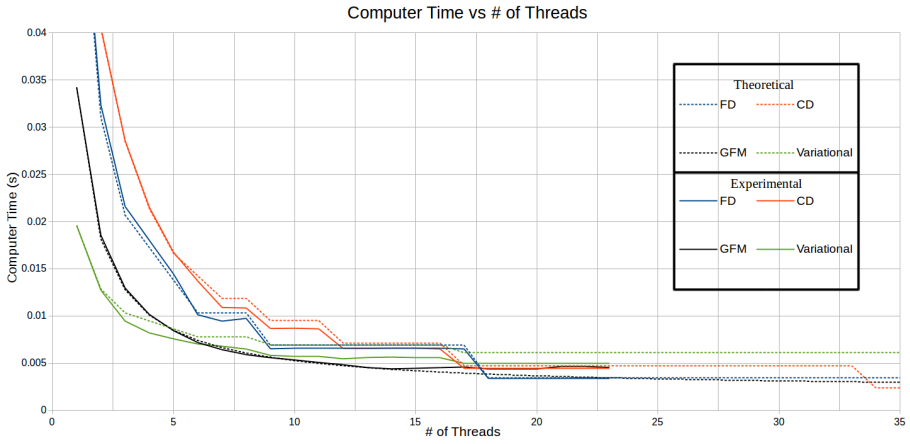


Figure 3.5: Experimental results demonstrating how our parallel version of the Variational approach scales with the number of threads. This is done for the Calcium Ion test problem. The sensitivities are approximated such that the error is around 10^{-4}.

We then repeat this experiment with the tolerances chosen to give a max error in

$y_p(t)$ around 10^{-6}. The results are shown in Figure 3.6. In this case, FD is unable to provide us with the requested accuracy, so it is not included. We again see that the variational equations perform the best in a sequential environment, but if enough threads are available, then the forward GFM and CD approaches would be preferred. Experimentally, we see that CD takes longer than expected as we increase the number of threads. The forward GFM approach again performs well, since there is a large amount of work for the threads to share.

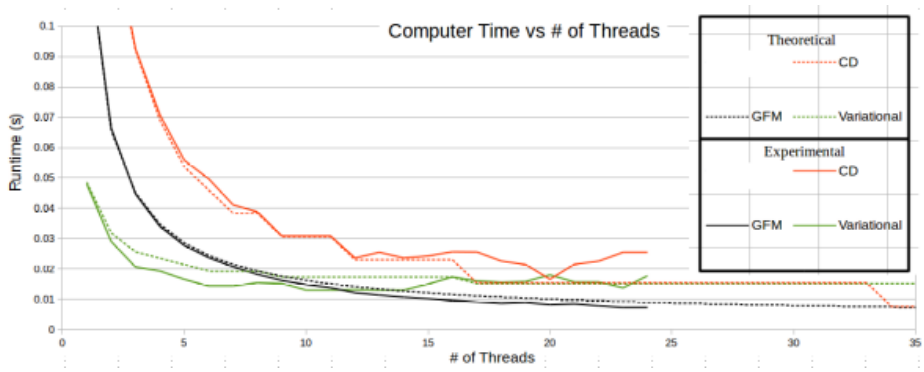


Figure 3.6: Experimental results demonstrating how our parallel version of the Variational approach scales with the number of threads. This is done for the Calcium Ion test problem. The sensitivities are approximated such that the error is around 10^{-6}.

In Table 3.10, we summarize the potential best performance of the methods based on the assumption that we could implement them in parallel without any overhead or communication costs associated with increasing the number of threads. The variational approach is limited by the fact that we have to simulate $y(t)$ along with the sensitivity for each parameter, $y_{p_k}(t)$, whereas forward GFM, FD, and CD are only limited by the time taken to perform the simulation of $y(t)$. In order to achieve the required accuracy in all components of $y_p(t)$, each approach requires a different level of accuracy in $y(t)$.

method	TOL $= 10^{-4}$		TOL $= 10^{-6}$	
	solver tolerance for $y(t)$	lower bound on cost (s)	solver tolerance for $y(t)$	lower bound on cost (s)
Variational	10^{-4}	0.005	10^{-6}	0.017
FD	10^{-8}	0.0033	–	–
GFM	10^{-5}	0.0028	10^{-7}	0.004
CD	10^{-6}	0.0022	10^{-9}	0.0077

Table 3.10: Comparison of lower bounds of expected costs for parallel sensitivity methods for the Calcium Ion test problem. (For GFM, we use strict defect control in the CRK solver when simulating $y(t)$, since we require accurate offmesh values of $y(t)$ in PMM.)

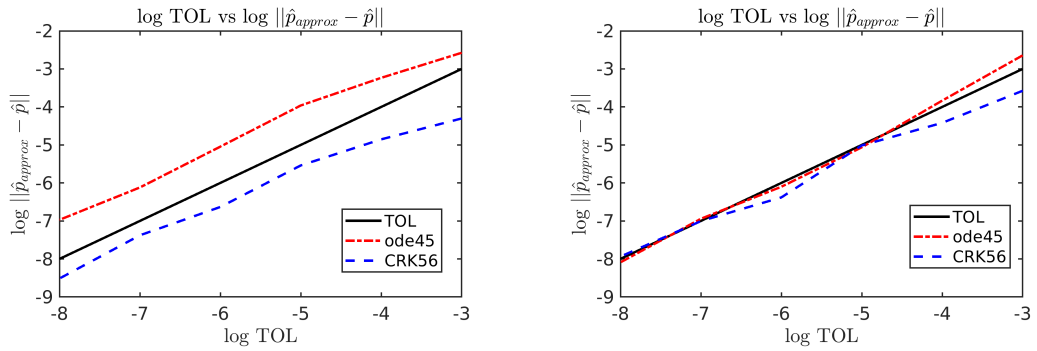
3.6 Full Optimization for ODEs

As we discussed in the Introduction, the objective of a numerical algorithm for solving this parameter estimation problem is to determine $\hat{p}_{approx}$, such that,

$$\|\hat{p} - \hat{p}_{approx}\| < K \text{ TOL},$$

for some user specified tolerance, TOL. We now demonstrate that one must be careful when choosing the tolerances used by the IVP solver and the optimizer. For this example, we consider the Barnes problem (Section 2.6.2). For varying values of TOL, we estimate the best fit parameters using two different IVP solvers and report the achieved accuracy (averaged over 5 data sets) in the left of Figure 3.7.

In both cases, we use a relative and absolute error tolerance of TOL for the IVP solver. The two solvers we consider are Matlab's ode45 and the CRK56 solver used in DDEM, with strict defect control. To perform the optimization, we use Matlab's implementation of LM in the routine lsqnonlin, with the StepTolerance chosen to be TOL. This sets the termination condition on the k^{th} iteration to be $\|p_k - p_{k-1}\| < \text{TOL}(\sqrt{\epsilon_{mach}} + \|p_k\|)$.



With Variational Equation Sensitivities With CD Sensitivities

Figure 3.7: Achieved accuracy vs Requested accuracy for the Barnes Problem

To approximate the required Jacobian, we simultaneously simulate the variational equations along with the original IVP. We found it necessary to specify a relaxed absolute tolerance on the variational equations, otherwise both ode45 and CRK56 would simulate the original IVP to more accuracy than the user requested. In this experiment, we set this relaxed tolerance to be $min(100\text{TOL}, 10^{-2})$. This ensures that we still have some accuracy in the variational equations, but it doesn't significantly impact the accuracy achieved in the approximation of the original IVP.

We observe that both ode45 and CRK56 give estimates that are consistent with TOL, but ode45 is consistently giving accuracy of around 10 TOL, whereas CRK56 is closer

Method	TOL					
	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}
FD	0.41 (0.05)	0.30 (0.04)	0.44 (0.07)	0.45 (0.04)	0.91 (0.08)	4.10 (0.35)
CD	0.41 (0.05)	0.30 (0.04)	0.60 (0.19)	0.35 (0.02)	0.31 (0.01)	0.47 (0.05)
Vari	0.09 (0.01)	0.21 (0.04)	0.25 (0.05)	0.22 (0.00)	0.25 (0.01)	0.42 (0.05)

Table 3.11: Average achieved accuracy in $\hat{p}_{approx}$ reported in units of TOL for the Barnes problem, with standard errors reported in brackets (averaged over 100 simulated data sets)

to the requested TOL. At relaxed tolerances, we see that CRK56 delivers too much accuracy. This is likely due to how we chose our minimum relaxed tolerance for the variational equations.

We also note that these results will change depending on the optimizer used and how any required sensitivities are approximated. For example, if we still use LM as the optimizer, but used centered divided differences to approximate the Jacobian, we obtain the results shown in the right of Figure 3.7. Here, ode45 and CRK56 both give accuracy consistent with the user specified TOL. Note that ode45 does better in this case than in the previous case, due to the fact that we have simulated all trajectories simultaneously when simulating the required centered difference trajectories. This is necessary in order to effectively vectorize the code in Matlab and it also results in more reliable sensitivities, since it ensures each trajectory is simulated using the same sequence of steps. CRK56 simulates each trajectory independently and still reliably meets the requested error tolerance in this example.

We next repeat this experiment, but only consider using CRK56 as the IVP solver. This is done to allow us to compare the methods in terms of both accuracy (Table 3.11) and computational cost (Table 3.12).

In terms of cost, we quantify the cost by dividing the total time taken by the time required to perform a single trajectory simulation with TOL=10^{-3}. As expected, the timing results here are consistent with the times required to approximate the model sensitivities. However, FD begins to take more time once it reaches its limit on how accurately it can approximate the model sensitivities.

Lastly, we consider similar results for the method of Ramsay (Section 2.5.3). In this case, we have used cubic B-splines with a knot at each observation time. The results are shown in Figure 3.8, for varying values of the smoothing parameter, λ. As λ increases, we see that the accuracy achieved by Ramsay's approach is determined by the ability of the B-splines to satisfy the IVP. This means that in order for $\hat{p}_{approx}$ to be accurate enough, we would have to take care in how we choose the order and knot placement of

Method	TOL					
	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}
FD	36	37	48	59	76	122
CD	51	52	70	89	116	137
Vari	27	26	34	43	57	69

Table 3.12: Average cost reported in units of time taken to simulate a single trajectory with TOL=10^{-3} to obtain $\hat{p}_{approx}$ for the Barnes problem (averaged over 100 simulated data sets)

the B-splines.

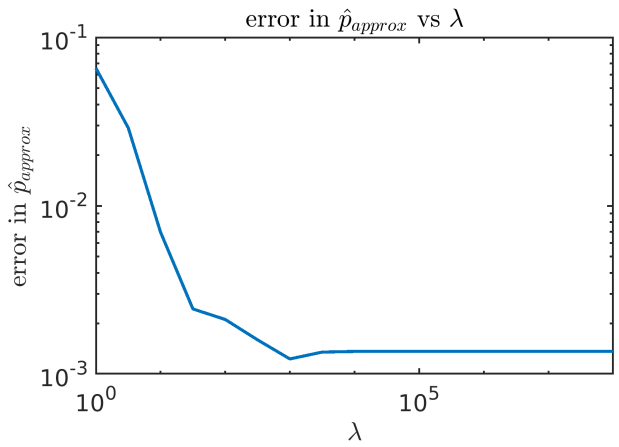


Figure 3.8: Achieved accuracy vs λ for Ramsay's method applied to the Barnes problem with $n_o = 11$

3.7 Extension to DDEs

We now briefly discuss how each of the above approaches can be applied to the case of constant lag DDEs.

Finite Differences

Finite differences can be directly applied to the case of DDEs, but one must be careful as the accuracy might be affected by the presence of discontinuities in the solution of the IVP and how accurately the DDE solver locates these discontinuities. The user also has to be sure to choose an appropriate value for ϵ_{FD} or ϵ_{CD}. For example, the DDEM package we consider locates discontinuities to within the user provided TOL, so the perturbations, ϵ_{FD} or ϵ_{CD}, should not be less than TOL.

Variational Equations

For constant lag DDEs, the variational approach results in the following neutral DDE, which we obtain by taking the time derivative of $\frac{\partial y}{\partial p}(t)$,

$$\begin{aligned}\frac{d}{dt}\frac{\partial y}{\partial p}(t) &= \frac{\partial}{\partial p}\frac{dy}{dt}(t)\\ &= \frac{\partial}{\partial p}f(t, y(t,p), y(t-\tau,p), p)\\ &= \frac{\partial f}{\partial y}(t)\frac{\partial y}{\partial p}(t) + \frac{\partial f}{\partial p}(t) + \frac{\partial f}{\partial \nu}(t)\frac{\partial y}{\partial p}(t-\tau) - y'(t-\tau)\frac{\partial \tau}{\partial p},\end{aligned}$$

where $\nu = y(t-\tau)$. This matrix valued DDE can be approximated simultaneously with the original system(1.4), with the initial conditions, $\frac{\partial y}{\partial p}(0)$, whose (i,j) entry is,

$$\frac{\partial y_i}{\partial p_j}(0) = \begin{cases} 1, & \text{if } p_j \text{ is the initial condition for } y_i \\ 0, & \text{otherwise} \end{cases}.$$

The corresponding history function is given by,

$$\frac{\partial y_i}{\partial p_j}(t) = \frac{\partial h_i}{\partial p_j}(t), \text{for } t < 0.$$

A limitation of this approach is that the variational system consists of $n_y + n_y n_p$ differential equations. This approach extends in a straightforward way to general systems of DDEs. For details, see [57]. This approach has previously been implemented in the DDEM [55] package, which makes use of the underlying CRK interpolant to efficiently simulate this system of neutral DDEs and appropriately handle any discontinuities that may arise.

If we want to consider using forward GFM for DDE IVPs, we need to be able to efficiently obtain offmesh values of $y_p(t)$. The forward approach we have described could be considered, although an offmesh evaluation of $y_p(t)$ would be rather expensive, as it would require us to store the mesh of $y_p(t)$ values used during our sequence of PMM steps and then apply (3.7) each time an offmesh value is needed.

3.7.1 Full Optimization for DDEs

As in the ODE case, we now demonstrate that one must be careful when choosing the tolerances used by the DDE IVP solver and the optimizer. For this example, we consider the

Kermack-McKendrick problem (Section 2.6.4). For varying values of TOL, we estimate the best fit parameters and report the achieved accuracy in Table 3.13 and the computational cost in Table 3.14. To perform the optimization, we use Matlab's implementation of LM in the routine lsqnonlin, with the StepTolerance chosen to be TOL.

Method	TOL			
	10^{-2}	10^{-3}	10^{-4}	10^{-5}
FD	1.69 (0.29)	0.47 (0.08)	0.68 (0.18)	0.44 (0.08)
CD	0.40 (0.09)	0.12 (0.04)	0.20 (0.05)	0.23 (0.07)
Vari	0.36 (0.10)	0.38 (0.15)	0.87 (0.33)	0.99 (0.35)
dde23 FD	0.84 (0.12)	0.34 (0.08)	0.35 (0.08)	0.25 (0.10)
ddesd FD	0.99 (0.41)	0.36 (0.12)	0.81 (0.11)	0.68 (0.09)

Table 3.13: Average achieved accuracy in $\hat{p}_{approx}$ reported in units of TOL for the Kermack-McKendrick problem, with standard errors reported in brackets (averaged over 20 simulated data sets)

Method	TOL				
	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
FD	24	38	53	106	131
CD	29	55	83	177	221
Vari	34	26	30	46	70
dde23 FD	57	329	2985	45848	-
ddesd FD	99	190	825	7151	-

Table 3.14: Average cost reported in units of time taken to simulate a single trajectory with TOL=10^{-2} (using the same DDE solver as was used in the optimization) to obtain $\hat{p}_{approx}$ for the Kermack-McKendrick problem (averaged over 20 simulated data sets)

The solvers we consider in these experiments are Matlab's dde23 and ddesd solvers, as well as the DDEM solver with strict defect control. We see that while dde23 can be successfully used as the DDE IVP solver, it is computationally expensive if much accuracy is required, since it is a second order method. The ddesd solver is a fourth order method, but we observe that it still becomes very expensive as the tolerance becomes more strict. Since we require reasonably accurate sensitivities, it is necessary to use fairly strict tolerances when applying FD and CD to approximate the model sensitivities. For this example, we found it necessary to use $\epsilon_{ODE} = \text{TOL}^2$ when simulating the FD and CD model trajectories. Unlike the Matlab solvers, using FD or CD with DDEM remains fairly inexpensive as we increase the accuracy requested.

For the variational approach, we used DDEM's implementation, with a relative and absolute error tolerance of TOL for all components. Performing the optimization using

the variational approach to approximate the sensitivities only becomes about twice as expensive if we go from TOL $= 10^{-2}$ to 10^{-6}. This increase is quite modest compared to the increase in cost when using FD or CD.

Chapter 4

Adjoint Method

We now turn our attention to the adjoint method, which is best suited for cases where the objective function of interest is scalar and the model consists of more parameters than state variables. We first consider how the method is applied in the case of ODE IVPs, then we go on to discuss how the method extends to DDE IVPs. In this chapter, our main focus is on how the cost of the adjoint method scales with the number of observations, n_o, in our least squares objective function. We investigate this dependence on n_o through numerical experiments and consider ways to reduce this cost.

4.1 Adjoint Method for ODE IVPs

For a more complete discussion of the adjoint method, we refer the reader to [11]. We present a full derivation of the adjoint method for ODEs in Appendix A.1. The derivation considers objective functions of the form,

$$G(y,p) = G(y(p)) = \int_0^T g(y(t,p))\,dt. \tag{4.1}$$

The adjoint method is defined by introducing the adjoint vector, $\lambda^T(t)$, which is the solution of the system of IVPs,

$$\dot{\lambda}^T(t) = \frac{\partial g}{\partial y}(t) - \lambda^T(t)\frac{\partial f}{\partial y}(t)\ ;\ \lambda^T(T) = 0, \tag{4.2}$$

where $\lambda^T(t)$ is of dimension n_y, $\dot{\lambda}$ denotes $\frac{d\lambda}{dt}$, and f comes from (1.1). The sensitivities of $G(y(p))$ are given by the equation,

$$\frac{\partial G}{\partial p} = -\int_0^T \lambda^T(t)\frac{\partial f}{\partial p}(t)\,dt - \lambda^T(0)\frac{\partial y}{\partial p}(0). \tag{4.3}$$

Note that the above equations are defined with t varying from T to 0. Defining $x = T - t$, we can return to the standard situation where the independent variable varies from 0 to T. For our purposes, there are two sensitivity equations of interest. The first corresponds to the model sensitivities and the second is the sensitivity of the NLS objective function (1.7).

4.1.1 Case of Model Sensitivities

In this case, we are considering the model sensitivity, $\frac{\partial y_j(t_i,p)}{\partial p}$, where $y_j(t_i,p)$ is the j^{th} component of the solution of the ODE at time t_i. In order to approximate this quantity using the adjoint method, we must set $T = t_i$ and $\lambda_j^T(t_i) = 1$. This corresponds to the objective function, $G(y,p) = y_j(t_i,p)$. While this is fine mathematically, it is inefficient to actually implement. The difficulty is that since $y(t)$ has n_y components, we must apply the adjoint approach to each component of $y(t)$. Moreover, if we want $\frac{dy}{dp}(t_i)$ for more than one t_i, then we must apply the adjoint approach at each t_i. This is because we are equivalently asking for the sensitivity of a vector valued objective function at each t_i, which is not what the adjoint method is efficient at computing. The variational approach discussed in Section 3.3 is better suited to this task, since it directly approximates the model sensitivities over the interval of interest.

4.1.2 Case of NLS Objective Function

We now make use of a different characterization of (1.7) that is consistent with the form of the objective function (A.1). We re-write $O(p) \equiv G(y(p))$ using the Dirac delta function as,

$$O(p) = \int_0^T g(y(t,p))\,dt = \int_0^T \Big[\sum_{i=1}^{n_o}\sum_{j=1}^{n_y} \frac{(\tilde{y}_j(t_i) - y_j(t_i,p))^2}{2}\delta(t_i - t)\Big]dt,$$

where $\delta(t - t_i)$ is the Dirac delta function, which is zero everywhere, except at $t = t_i$. $\delta(t - t_i)$ has the property that,

$$\int_a^b q(t)\delta(t - t_i)dt = q(t_i), \tag{4.4}$$

for any sufficiently smooth function $q(s)$, if $a < t_i < b$. With this representation of g,

$$\frac{\partial g}{\partial y} = \sum_{i=1}^{n_o} (y(t_i, p) - \tilde{y}(t_i))\delta(t - t_i).$$

In this case, we must evaluate the $\sum_i (\tilde{y}(t_i) - y(t_i, p))\delta(t - t_i)$ term accurately when approximating the solution of (4.2). The presence of this term will lead to discontinuities in $\lambda^T(t)$, whenever $t = t_i$. The natural way to account for these discontinuities is to approximate the solution of (4.2) on each subinterval (t_i, t_{i+1}), and apply the jump,

$$\lambda^T(t_i)^- = \lambda^T(t_i)^+ + (\tilde{y}(t_i) - y(t_i, p)). \tag{4.5}$$

to obtain the initial conditions for the next subinterval (t_{i-1}, t_i). It is having to restart the solver at each observation time, t_i, to apply this jump condition that makes the cost of the adjoint approach particularly sensitive to n_o. We will now briefly review how the adjoint method extends to constant lag DDEs.

4.1.3 Adjoint Method for constant lag DDE IVPs

We consider here the special case of constant lag DDEs with a delay of the form $\alpha = t - \tau$ and constant history function, $y(t) = y_o$, for $t < 0$. For simplicity, we assume there is only one delay, but the analysis and techniques extend in a straightforward way to the case of multiple delays. For a rigorous derivation of the adjoint method for more general systems of DDEs, we refer the reader to [56]. We present a derivation of the adjoint method for constant lag DDE IVPs in Appendix A.2. The difference here compared to the ODE case is that f not only depends on $y(t)$, but also on $y(t - \tau)$. For convenience, we let $\nu = y(t - \tau)$. Similar to the ODE case, the adjoint system for this constant lag DDE is defined by requiring that an associated adjoint vector, $\lambda^T(t)$, be the solution of the system of IVPs,

$$\dot{\lambda}^T(t) = \frac{\partial g}{\partial y}(t) - \lambda^T(t)\frac{\partial f}{\partial y}(t) - \lambda^T(t + \tau)\frac{\partial f}{\partial \nu}(t + \tau)\,;\lambda^T(t) = 0, \text{ for } t \geq T. \tag{4.6}$$

The sensitivities are then given by,

$$\begin{aligned}\frac{\partial G}{\partial p} &= -\int_0^T \lambda^T(t)\Big(\frac{\partial f}{\partial \nu}(t)y'(t - \tau)\frac{\partial \alpha}{\partial p}(t) + \frac{\partial f}{\partial p}(t)\Big)\,dt \\ &\quad -\lambda^T(0)\frac{\partial y}{\partial p}(0) - \int_{-\tau}^0 \lambda^T(t + \tau)\frac{\partial f}{\partial \nu}(t + \tau)\frac{\partial h}{\partial p}(t)\,dt.\end{aligned}$$

where $h(t)$ is the history function.

4.1.4 Additional Considerations for DDEs

For DDEs, we have to be careful to properly handle discontinuities and store accurate approximations of the solution over previous subintervals. As in the ODE case, we have to restart the IVP solver at each observation point, t_i, when integrating the adjoint system of IVPs from T to 0. However, the discontinuities introduced by (4.5) at each t_i will be encountered again, since $\dot{\lambda}^T(t)$ depends on the lagged value, $\lambda^T(t+\tau)$. Also, discontinuities in $y(t)$ and its derivatives must be taken into consideration as well. It is usually the case that $y(t)$ will be continuous, but its higher derivatives may not be. For example, if the history is constant, $h(t) = y_0$, then the derivative to the left of the initial time is zero, while it is $f(t, y_0, p)$ to the right of the initial time. For a fixed lag, $t - \tau$, we will encounter this discontinuity again when $t = \tau$. At this point, the discontinuity is propagated, but in a derivative of order one higher. Eventually the propagated discontinuity is of sufficiently high order that it can be ignored. This will be the case when the order of the derivative discontinuity is higher than the order of the underlying CRK formula used to approximate $\lambda^T(t)$.

To handle these additional discontinuities, we must restart the solver at each of them. In Figure 4.1, we illustrate the impact that the discontinuities have on the behaviour of the solution of the DDE and its associated adjoint system. As we will see, the presence of these discontinuities makes the adjoint approach significantly more expensive for the case of DDEs than it is for ODEs. For a detailed discussion of how discontinuities in DDEs impact the smoothness of the sensitivities, we refer the reader to [1].

4.1.5 Using the Adjoint Method in an LM optimizer

For the LM algorithm, we require the gradient of each term in the least squares objective function. That is, we need a Jacobian matrix, not just a gradient vector. In such cases, we have to apply the adjoint method once for each term in the objective function (i.e. each row of the Jacobian). This leads us to the observation that we can group terms to reduce the number of times we have to apply the adjoint method.

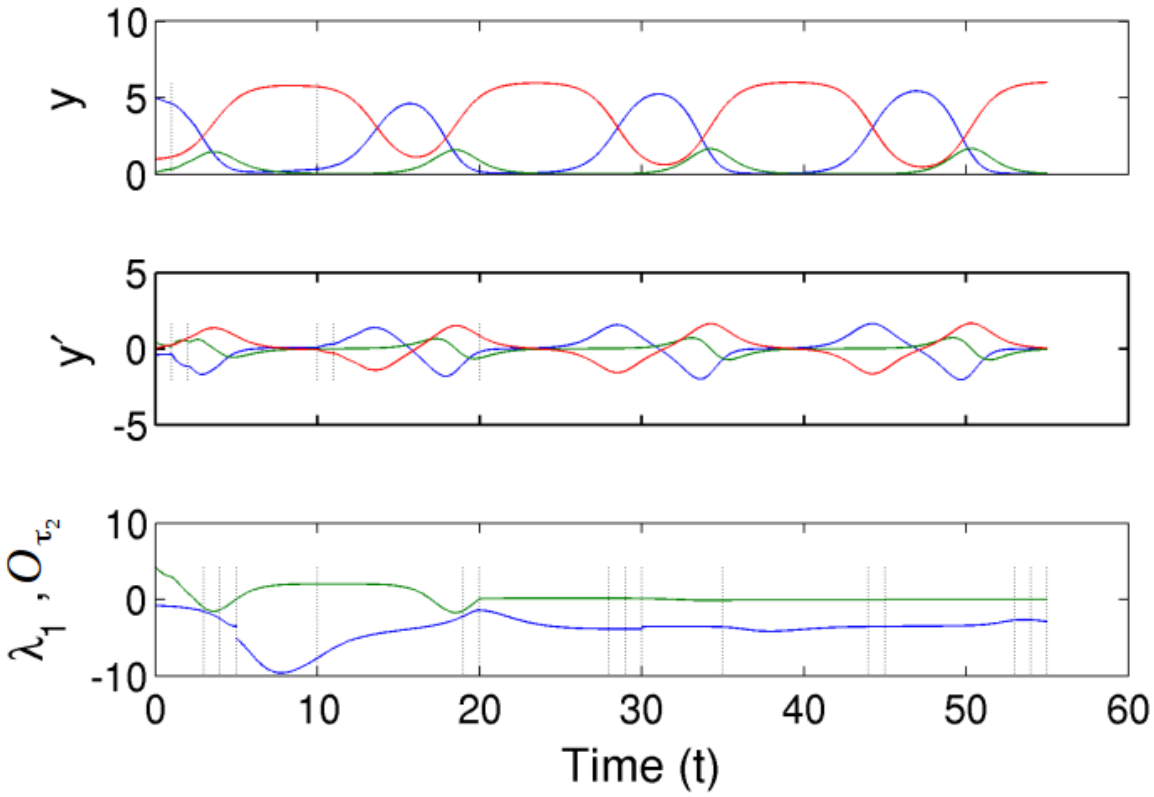


Figure 4.1: Top: Solution of each component of the Kermack-McKendrick model (see section 2.6.4). Middle: Derivative of each component of the solution of the model. Bottom: The first component of the adjoint vector, $\lambda_1^T(t)$ (blue), and the cumulative value of the sensitivity integral, $\frac{\partial O}{\partial \tau_2}(t)$ (green). The locations of discontinuities, up to second order, are indicated by dotted lines at the times where they occur. For this model, $\tau_1 = 1$, and $\tau_2 = 10$. Observations are at times 5,30, and 55.

Grouping Observations

To aid with the explanation, we adopt slightly different notation for the NLS objective function (1.7), namely,

$$O(p) = \frac{1}{2}\sum_{i=1}^{n_o}\sum_{j=1}^{n_y} r_{ij}^2, \tag{4.7}$$

where $r_{ij} = y_j(t_i, p) - \bar{y}_j(t_i)$. Instead, we rewrite our objective function as,

$$O(p) = \frac{1}{2}\sum_{i=1}^{n_o} d_i^2, \tag{4.8}$$

where $d_i = \sqrt{\sum_{j=1}^{n_y} r_{ij}^2} = \|r_i\|_2$. More generally, we could consider forming different d_i's,

which sum over not only components of y, but also over subsets of observation times. If we group these d_i's into n_k groups, our objective function becomes,

$$O(p) = \frac{1}{2}\sum_{k=1}^{n_k} e_k^2, \tag{4.9}$$

where,

$$e_k = \sqrt{\sum_{i \in I(k)} d_i^2}.$$

Here, each $I(k)$ is a disjoint set of indices, such that,

$$\cup_{k=1}^{n_k} I(k) = \{1, \ldots, n_o\}.$$

While mathematically equivalent, it turns out that when we attempt to optimize (4.7) or (4.8), we observe different performance. To see this, in the next section, we consider using Levenberg-Marquardt (or Gauss-Newton) to solve this non-linear least squares problem.

Numerical Experiment

As an example, we have taken $y(t, p)$ to be a simple non-linear function and used Matlab's implementation of Levenberg-Marquardt to perform the optimization. The results shown in Table 4.1 are for one set of noisy observations, with all optimizations started from the same initial guess. In this experiment, we have chosen the groups such that each $I_k = \{k, k + n_k, k + 2n_k, \ldots\}$.

For this example, we use the more general form (4.9) and see how varying the number of groups impacts the performance of the optimizations. For this example, we have 20 observation times. We quantify the cost by,

$$\text{normalized cost} = \frac{(\#\text{ of groups})(\#\text{ of iterations})}{(20)(\#\text{ of iterations without grouping})}. \tag{4.10}$$

We observe that the required number of iterations increases as we use fewer groups and the number of iterations can vary dramatically, depending on how the observations are grouped. This appears to be due to which observations happen to get grouped together. For example, when we use 12 groups we required over 5 times as many iterations as when we used 20 groups. This suggests we have to be careful when grouping observations. Also, we note that if we group the observations into too few groups (i.e. $n_k \leq n_p$, with $n_p = 2$ in this example) then the method converges very slowly). Again, these results are just for one simple non-linear model, so some of the behaviour observed may be specific

# of groups	normalized # of iterations	normalized cost
20	1	1
19	1.2	1.14
18	1.4	1.26
17	1.5	1.275
16	2.2	1.76
15	2.1	1.575
14	3.3	2.31
13	3.9	2.535
12	5.7	3.42
11	2.8	1.54
10	3.2	1.6
9	2.7	1.215
8	2.8	1.12
7	3.5	1.225
6	3	0.9
5	4.5	1.125
4	6	1.2
3	6.5	0.975
2	10	1

Table 4.1: Experimental results demonstrating how grouping observations can impact the performance of the optimizer.

to this problem. These results suggest that the additional number of iterations required by the optimizer is roughly proportional to the reduction in the number of times the adjoint method is applied per iteration, although the number of iterations is sometimes significantly higher.

Also, if we look at the progress made by the optimizer (results not shown here), regardless of how many groups are used, the first few iterations quickly reduce the residual, but then, for some groupings of observations, the optimizer becomes very slow and begins to follow the steepest descent direction on each iteration (the LM damping parameter, λ, does not go to zero). This seems to indicate that the J^TJ approximation to the Hessian used in Levenberg-Marquardt is very poor for some groupings of observations. We will now briefly discuss how grouping observations impacts the Hessian approximation.

Hessian Approximation

As we know, LM relies on J^TJ being a reasonable approximation of the Hessian of O. In the following, we'll consider (4.9), with $n_y = 1$. This simplifies things, so that $d_i = r_i$. If we don't group observations, we have that the (j, l) entries of the Hessian of (4.7) are

given by,

$$H_{jl} = \sum_{i=1}^{n_o} (\frac{\partial r_i}{\partial p_j}\frac{\partial r_i}{\partial p_l} + r_i \frac{\partial^2 r_i}{\partial p_j \partial p_l}). \tag{4.11}$$

If we group observations, we instead have,

$$H_{jl} = \sum_{k=1}^{n_k} (\frac{\partial e_k}{\partial p_j}\frac{\partial e_k}{\partial p_l} + e_k \frac{\partial^2 e_k}{\partial p_j \partial p_l}). \tag{4.12}$$

Under the assumption that each residual, r_i is small, we have that $H \approx J^T J$ when we don't group observations. However, when we group observations, we need each e_k to be small in order to ignore the second term. Since $e_k = \sqrt{\sum_{i \in I(k)} r_i^2}$, this assumption is less likely to hold. This means that our approximation to the Hessian is less likely to be accurate when we optimize (4.9). We suspect this is why the optimizer requires more iterations to converge when more observations are grouped together in our numerical experiment above.

4.1.6 Parallel Adjoint Method

We can divide the work based on subsets of the observations (similar to the discussion of grouping observations in section 4.1.5). Specifically, we can express the objective function as the sum of several objective functions, each of which will have its sensitivity computed independently. This would not help us much with the variational approach, since the main component of the cost is in performing the simulation, not evaluating the piecewise polynomial approximation to the model sensitivities at each observation time. However, this could be useful in the case of the adjoint method, since the time required to approximate the adjoint IVP may be reduced if we divide the observations between processors. For example, consider the case of uniform observations and two processors (P^1 and P^2). We could rewrite the NLS objective function as, $O(p) = O^{(1)}(p) + O^{(2)}(p)$, where

$$O^{(1)}(p) = \sum_{\text{odd i}} \frac{\|\tilde{y}(t_i) - y(t_i, p)\|^2}{2}, \text{ and } O^{(2)}(p) = \sum_{\text{even i}} \frac{\|\tilde{y}(t_i) - y(t_i, p)\|^2}{2},$$

for $i = 1, \ldots, n_o$, where we have divided the observations so that $O^{(1)}(p)$ contains the odd observations and $O^{(2)}(p)$ contains the even observations. By linearity, we have that

$$\frac{\partial O}{\partial p} = \frac{\partial O^{(1)}}{\partial p} + \frac{\partial O^{(2)}}{\partial p}. \tag{4.13}$$

This means that we can have P^1 compute $\frac{\partial O^{(1)}}{\partial p}$, while P^2 independently computes $\frac{\partial O^{(2)}}{\partial p}$. The only communication necessary is to evaluate the sum to obtain $\frac{\partial O}{\partial p}$. Since the observations are evenly spaced, say a distance Δt apart, then each observation for a given processor would now be $2\Delta t$ away from the neighbouring observations. We would still have to perform the forward simulation of the original IVP on both processors, but for the reverse simulation of the adjoint IVP, each processor will only have half as many observations that it is forced to restart the simulation at. This will have the most beneficial effect on reducing the cost in cases where the space between observations is restricting the step size, which would result in half as many steps being taken (per processor) as in the single processor case, hence a potential speedup by a factor of 2 (for the approximation of the adjoint IVP part of the computation).

To demonstrate this, we have implemented this approach for the Mendes model (see section 2.6.6), with the parallelism implemented using MPI. We consider a case where we have 2000 observation times and expect to see significant speedup. The results are shown in Figure 4.2. We see that the Adjoint method scales well in this case, since we have a large number of observations.

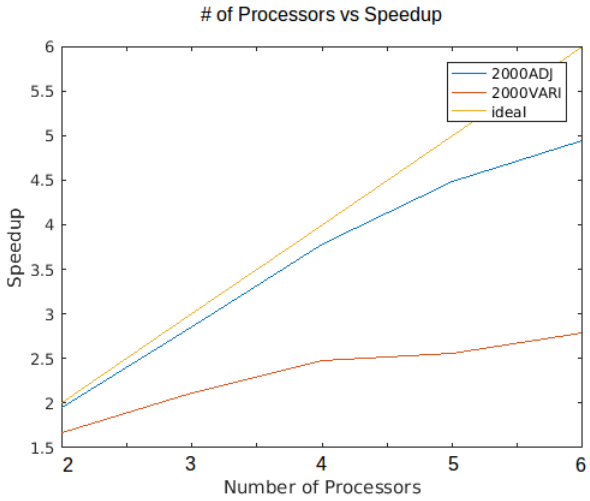


Figure 4.2: Experimental results for the Mendes test problem demonstrating how our parallel version of the adjoint method scales with the number of processors. As reference, we also show how the variational approach (bottom line) scales. The forward trajectory and variational equations are simulated using the sixth order CRK solver used in DDEM. For simulation of the adjoint IVP, we are using an RK solver, for reasons discussed in section 4.2.3.

4.2 Numerical Experiments

4.2.1 Determining the Order of Discontinuities to Track in the adjoint DDEs

First, we investigate what order of discontinuity it is necessary to track, in order to obtain reasonable performance when approximating the adjoint DDE. Note that since we use reliable error control when approximating the adjoint IVP, even if we only track the jump discontinuities occurring at each of our observations, we should still obtain accurate sensitivities. This is illustrated in Table 4.2. However, we see that we end up either taking extra steps in order to satisfy the error requirements or we take extra steps when we force the integration to stop at locations of all the identified discontinuities, which are not based on controlling the local truncation error of the adjoint IVP. Given these results, we only track discontinuities in the solution of the adjoint DDE and its first and second derivatives for the remainder of our experiments. (The resulting error control for the adjoint IVP will be less reliable but adequate for most problems.)

4.2.2 Dependence on n_o

We also investigate how many observations we can have before the cost of the adjoint approach is prohibitively expensive, relative to that associated with solving the variational equations. For the case of ODEs, we consider the Barnes problem (Section 2.6.2). As we can see in Figure 4.3, the adjoint approach requires less computer time than the variational approach up to around 400 observations. We also note that for small numbers of observations, the adjoint method performs fairly consistently, up until the spacing of the observations begins to restrict the step size the solver is able to take. For the variational approach, the cost remains flat, since increasing n_o only increases the number of off-mesh interpolations we have to make, which is much cheaper than the cost of simulating the variational equations.

For DDEs, we consider the Kermack-McKendrick model. As we see in Figure 4.4, the cost of the adjoint approach depends strongly on the number of observations. For example, with TOL $= 10^{-5}$, the adjoint method is already more expensive than the variational approach when there are more than 6 observations.

Table 4.2: For several tolerances and maximum orders of discontinuity to track, we report the computer time taken by the adjoint method (including the time required for approximating the forward IVP) and the number of steps taken during the simulation of the adjoint IVP. This is done for the Kermack-McKendrick test problem (Section 2.6.4), with $n_o = 5$.

Max Order	TOL	Max Rel Error	Time	Adjoint Steps
0	0.001	0.00704	0.0144	106
1	0.001	0.00678	0.0135	**101**
2	0.001	0.00705	0.0131	106
3	0.001	0.00702	0.0145	127
4	0.001	0.00695	0.0168	159
5	0.001	0.0077	0.0183	182
6	0.001	0.00733	0.0197	204
7	0.001	0.00764	0.0214	231
8	0.001	0.00748	0.023	257
0	0.0001	0.000236	0.0228	165
1	0.0001	0.000236	0.0179	135
2	0.0001	0.000236	0.0163	**131**
3	0.0001	0.000236	0.0165	144
4	0.0001	0.000236	0.0176	167
5	0.0001	0.000236	0.0194	191
6	0.0001	0.000236	0.0206	213
7	0.0001	0.000236	0.0223	239
8	0.0001	0.000236	0.024	265
0	1e-05	5.39e-05	0.0369	282
1	1e-05	5.39e-05	0.0223	182
2	1e-05	5.39e-05	0.0186	**163**
3	1e-05	5.39e-05	0.019	175
4	1e-05	5.39e-05	0.0206	200
5	1e-05	5.39e-05	0.0223	224
6	1e-05	5.39e-05	0.0233	243
7	1e-05	5.39e-05	0.0244	263
8	1e-05	5.39e-05	0.0255	283

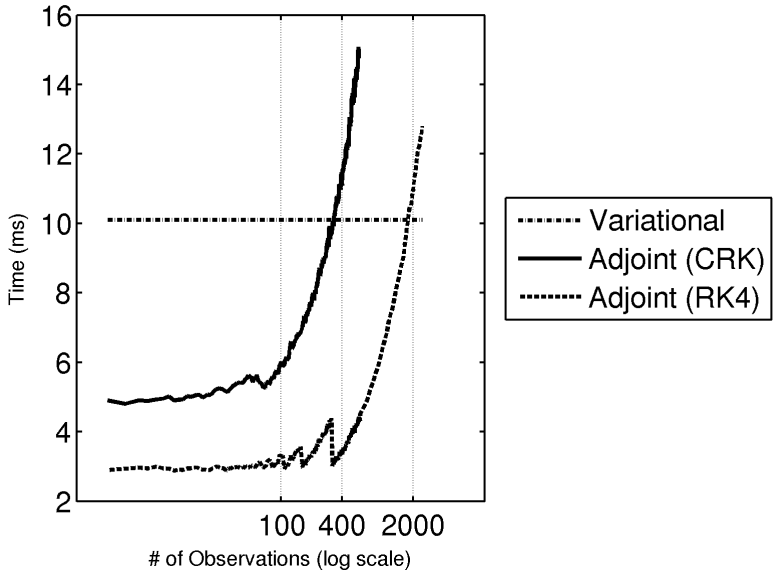


Figure 4.3: For TOL $= 10^{-6}$, we plot the time taken by the adjoint and variational approaches versus the number of observations for the Barnes ODE. We also show how using a fourth order RK method with no error control for the adjoint ODE can reduce the computer time.

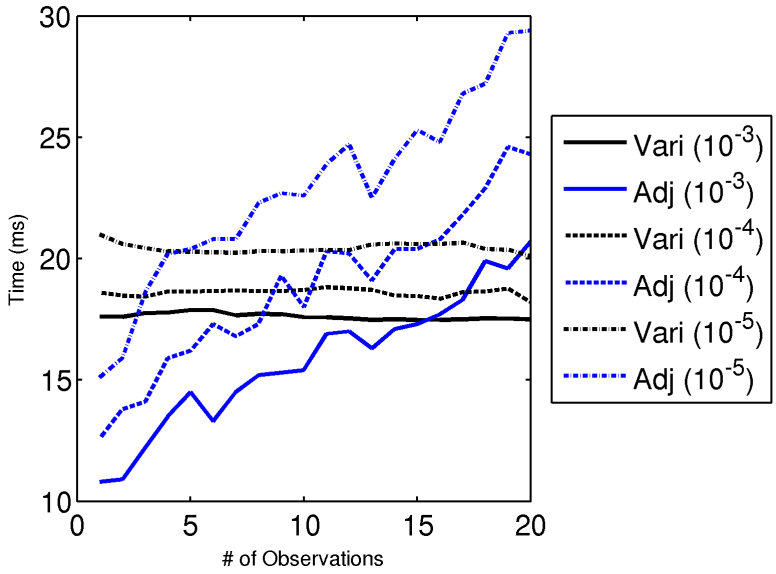


Figure 4.4: For different tolerances, we plot the time taken by the adjoint (blue) and variational (black) approaches versus the number of observations for the Kermack-McKendrick DDE.

4.2.3 Low Order Method for Approximating the Adjoint IVP

As discussed above, if we have a large number of observations, then our step size during solution of the adjoint equation might be severely restricted by having to restart the integration at each observation point rather than by only having to ensure the numerical accuracy of the solution be obtained. In such cases, it might be more efficient to use a lower order RK method for the adjoint ODEs or a lower order CRK method for the adjoint DDEs. As an example, for the ODE model, we have applied a fixed step size fourth order RK method (denoted RK4), with no associated error control, to the associated adjoint IVP. The impact on the cost is shown in Figure 4.3. We see that by using the lower order solver for the adjoint ODE, we are able to handle around 2400 observations before the adjoint method requires more computer time than the variational approach. On the other hand, the accuracy and reliability of the error in the approximate solution of the adjoint IVP will be reduced.

4.2.4 Continuous Objective Function Approximation

We now consider reducing the adjoint method's dependence on n_o by approximating our discrete NLS objective function with a continuous function. By doing so, we will no longer have discontinuities being introduced at each observation, but we will be introducing approximation error. Recall that our NLS objective function can be expressed as a sum of δ-functions. One definition of the δ-function is,

$$\delta(t_i - t) = \lim_{\sigma \to 0} \frac{1}{\sigma\sqrt{\pi}} e^{-\left(\frac{t-t_i}{\sigma}\right)^2}. \tag{4.14}$$

If we refer to this definition and choose a non-zero value of σ, then we obtain a continuous approximation to our objective function, which may lead to reasonable performance. However, if σ is too small, then we have no benefit, since the continuous objective function will still contain sharp peaks at each observation - making the adjoint system expensive to simulate. We also have to keep in mind that if σ is too large, then we may introduce additional errors.

A potential limitation of this approach is that the adjoint system and the sensitivity integrals depend on the products of $\lambda^T(t)$ with $f_y(t)$ and $f_p(t)$. So when we modify our objective function, we could introduce potentially large errors depending on the behaviour of $f_y(t)$ and $f_p(t)$. Also, we still have to compute the continuous approximation during each derivative evaluation for the adjoint system. We consider applying this approach to the Kermack-McKendrick test problem with both a varying number of observations and

a varying σ (results not shown). We observe that as σ increases, the runtime decreases and the error in the approximation of $y_p(t)$ increases. The cost of the adjoint method is less sensitive to n_o when this continuous approximation is used, but at the cost of less accuracy in the sensitivities.

Chapter 5

Handling Unobserved State Variables

While the methods discussed in Chapter 2 for obtaining a suitable initial guess for the model parameters can work well in the case where all state variables are observed, modifications are required when some of the state variables are unobserved. In this chapter, we consider the case where a subset of the state vector is unobserved. Specifically, we consider the case where $y = [y_{unob}, y_{ob}]$, where y_{unob} denotes the unobserved components and y_{ob} denotes the observed components of the state vector. We assume that all components of $y_{ob}(t)$ are observed at the same set of observation times.

5.1 Methods

Dattner [13] has proposed a method for estimating linear parameters in ODEs when only a subset of the state variables are observed. However, his proposed method requires that initial guesses be provided for the values of an artificial parameter vector, α, introduced to parameterize curves approximating the trajectories of the unobserved states. The approach of Ramsay discussed in Section 2.5.3 has a similar limitation. We will now briefly describe the approach taken by Dattner, then describe how we attempt to address this difficulty. We go on to propose two other related techniques and present results demonstrating the performance of these approaches on six test problems from the literature.

5.1.1 Dattner's Approach

First, we review Dattner's proposed approach. We consider the IVP (1.1). We further restrict f to be linear in p (and only depending on t through $y(t)$), so we can express f as, $f(t, y, p) = g(y(t))p$, where $g(y(t))$ is a matrix-valued function of dimension $n_y \times n_p$. In this case, the integral form of the solution of the IVP (1.1) is,

$$y(t) = y_o + \left[\int_0^t g(y(s))ds\right] p. \tag{5.1}$$

Now, given our observations at n_o discrete times $(\bar{y}(t_i), i = 1, \ldots, n_o)$, we construct an approximation to $y(t)$, call it $\tilde{y}(t)$, over the interval. This can be done using local polynomials or splines, for example. We then define our associated objective function to be,

$$M(p) = \int_0^T \left\| \tilde{y}(t) - y_o - \left[\int_0^t g(\tilde{y}(s))ds\right] p \right\|^2 dt, \tag{5.2}$$

which simply states that $\tilde{y}(t)$ should satisfy (5.1), $\forall t \in [0, T]$. Note that since this objective function is linear in p and y_o, minimizing it is a linear least squares problem. We denote the minimizer by $\hat{p}$ and the corresponding vector of initial conditions by $\hat{y}_o$. In terms of computation, this involves solving several linear systems and computing several quadratures involving functions of $\tilde{y}$ (see [13] for the precise formulae). Note, this is the same objective function used in INT-SME (2.9), but with the assumption that all parameters appear linearly ($q = \varnothing$).

In order to apply (5.2) in the case where only a subset of the state vector is observed, we have to introduce a technique to approximate $y_{unob}(t)$. The approach taken by Dattner is to define parameterized curves, $\tilde{y}_{unob}(\alpha, t)$, for some parameter vector, α. To determine a suitable α, recall that, for a given $\tilde{y}(t)$, we can obtain $\hat{p}$ and $\hat{y}_o$. For a given α, we denote these as $\hat{p}(\alpha))$ and $\hat{y}_o(\alpha)$. With this in mind, the objective function proposed by Dattner is,

$$M(\alpha) = \int_0^T \left\| \tilde{y}(\alpha, t) - \hat{y}_o(\alpha) - \left[\int_0^t g(\tilde{y}(\alpha, s))ds\right] \hat{p}(\alpha) \right\|^2 dt. \tag{5.3}$$

We note that this approach can also be done for SME instead of INT-SME. This would result in,

$$M(\alpha) = \int_0^T \left\| \tilde{y}'(\alpha, t) - g(\tilde{y}(\alpha, s))\hat{p}_{\text{SME}}(\alpha) \right\|^2 dt. \tag{5.4}$$

5.1.2 Our data driven simulation to approximate y_{unob}

In [13], Dattner discusses how the choice of basis functions used to approximate each of the unknown states requires knowledge of the expected behaviour of the solution. To address this, we suggest that the most natural choice should be a form that is designed to obey the required dynamics (i.e. the ODE itself). In general, we can define $\tilde{y}_{unob}(\alpha, t)$ to be the solution of the data driven ODE,

$$\tilde{y}'_{unob}(t) = g_{unob}(\tilde{y}(t))\alpha, \tag{5.5}$$

where g_{unob} contains the rows of g associated with the unobserved states. In this case, α is the subset of parameters that are associated with g_{unob}. While this means we have to solve a system of ODEs to obtain each unobserved state, we note that it is only a system of size equal to the number of unobserved states. Furthermore, in many applications we are only interested in obtaining crude initial guesses and we can use a lower order ODE solver to reduce the cost. For example, one might use a first order Euler approximation. With this choice for $\tilde{y}_{unob}(\alpha, t)$, note that we still need to provide an initial guess for α. Since α is a subset of p, we don't necessarily have to actually compute $\hat{p}(\alpha)$ in (5.3). Splitting the state into its observed and unobserved parts, we can write (5.3) as,

$$\begin{aligned} M(\alpha) = &\int_0^T \left\| \tilde{y}_{unob}(\alpha, t) - y_{unob}(0) - \left[\int_0^t g_{unob}(\tilde{y}(\alpha, s))ds \right] \hat{p}(\alpha) \right\|^2 dt \\ &+ \int_0^T \left\| \tilde{y}_{ob}(\alpha, t) - y_{ob}(0) - \left[\int_0^t g_{ob}(\tilde{y}(\alpha, s))ds \right] \hat{p}(\alpha) \right\|^2 dt. \end{aligned} \tag{5.6}$$

Let $p = [\alpha\ p_{ob}]$, where p_{ob} denotes the parameters that appear only in f_{ob} (and hence not in α). Then, let $\tilde{p}(\alpha) = [\alpha\ \hat{p}_{ob}]$. Replacing $\hat{p}(\alpha)$ with $\tilde{p}(\alpha)$ in (5.6),

$$\begin{aligned} \tilde{M}(\alpha) &= \int_0^T \left\| \tilde{y}_{unob}(\alpha, t) - y_{unob}(0) - \left[\int_0^t g_{unob}(\tilde{y}(\alpha, s))ds \right] \alpha \right\|^2 dt \\ &\quad + \int_0^T \left\| \tilde{y}_{ob}(\alpha, t) - y_{ob}(0) - \left[\int_0^t g_{ob}(\tilde{y}(\alpha, s))ds \right] \tilde{p}(\alpha) \right\|^2 dt \\ &= \int_0^T \left\| \tilde{y}_{ob}(\alpha, t) - y_{ob}(0) - \left[\int_0^t g_{ob}(\tilde{y}(\alpha, s))ds \right] \tilde{p}(\alpha) \right\|^2 dt, \end{aligned} \tag{5.7}$$

where the first term is taken to be zero, due to our choice that $\tilde{y}_{unob}(\alpha, t)$ satisfies (5.5). Of course, the main downside of this data driven approach is that we have returned to the

original problem - namely that we need initial guesses for some of the model parameters. As noted, this difficulty may be slightly reduced, depending on how many parameters are in α and how many are in p_{ob}. We now turn our attention to two alternative techniques, in which we approximate the unobserved states by using the structure of the model and the observed states.

5.1.3 Two Alternative Techniques

Given the observed state, we can approximate $y'_{ob}(t)$, as is done when we apply DBE or SME. In the following, we assume that we can rewrite $y'_{ob}(t) = f_{ob}(t, y, p)$ as,

$$y_{unob}(t) = h(t, y_{ob}(t), y'_{ob}(t), p). \tag{5.8}$$

That is, we can express $y_{unob}(t)$ as an explicit function of the observed quantities and the parameters. For example, this should usually be possible whenever $y_{unob}(t)$ appears linearly in f_{ob}.However, if too few components of the state vector are observed, this might not be possible.

Now, since we have an expression for $\tilde{y}_{unob}(t)$, we can either apply SME or INT-SME. However, we note that using (5.8) implies that the SME objective function will be zero for the observed components. Specifically, the first technique considers,

$$h'(t, y_{ob}(t), y'_{ob}(t), p) = f_{unob}(t, [h(t, y_{ob}(t), y'_{ob}(t), p),\ y_{ob}(t)], p). \tag{5.9}$$

While this approach is appealing due to its simplicity - approximating derivatives and solving one least squares problem - it is limited by the fact that it requires one to approximate the second derivative of the observed states. Depending on the nature of the data, this may be difficult to do accurately. Since SME can not estimate initial conditions, one can use observations near $t = 0$ to estimate $y_{ob}(0)$ and substitute the parameter estimates from (5.9) into (5.8) to estimate $y_{unob}(0)$. We will refer to this first technique as algrebraic smooth and match estimator (ASME). We note that this technique was originally proposed by Varah [49] and later appeared as a method for estimating parameters in second order ODEs [46].

The second technique considers,

$$h(t, y_{ob}(t), y'_{ob}(t), p) = y_{unob}(0) + \int_0^t f_{unob}(\tau, [h(t, y_{ob}(\tau), y'_{ob}(\tau), p),\ y_{ob}(\tau)], p)\, d\tau, \tag{5.10}$$

$$y_{ob}(t) = y_{ob}(0) + \int_0^t y'_{ob}(\tau)\, d\tau, \tag{5.11}$$

where the second equation may be used to estimate the initial conditions, $y_{ob}(0)$, or nearby observations can be used instead. Unlike ASME, this approach only requires us to approximate the first derivative of the observed states. It also requires an integral to be approximated, but this is likely going to be less problematic than approximating the second derivative. We will refer to this second technique as algrebraic integral smooth and match estimator (AINT-SME).

The main limitation of both ASME and AINT-SME is that they rely on the assumption (5.8), which will depend on the structure of the model. We will also see that they may give poor initial guesses if the model is not consistent with the observed data. One potential advantage is that, depending on the structure of the model, the resulting least squares problems might in fact be able to be expressed in the form of a linear least squares problem, with nonlinear constraints. This means that it might be possible to generate an initial guess by first solving the linear problem, without enforcing the constraints, then use this initial guess to solve the constrained problem. To demonstrate these approaches, we now consider several examples.

5.2 Numerical Examples

5.2.1 SIR measles model example

As a real world example, we consider fitting a basic SIR model to weekly case reports of measles in England and Wales from 1948 - 1949, as was done in [13]. The SIR model is given by,

$$S'(t) = -\beta S(t)I(t), \tag{5.12}$$

$$I'(t) = \beta S(t)I(t) - \gamma I(t), \tag{5.13}$$

$$R'(t) = \gamma I(t), \tag{5.14}$$

where $\gamma = 1.4$ is known. $S(t)$, $I(t)$, and $R(t)$ denote the susceptible, infectious, and recovered populations, respectively. The parameters to be estimated are $S(0), I(0),$ and β. $R(0)$ is assumed to be zero. Only $I(t)$ is observed once every time unit (one week), from $t = 0$ to $t = T = 52$. We denote the observed values by, $\bar{I}(t_i), i = 0, \ldots, 52$. The data is shown in Figure 5.1. As a starting point, we briefly consider how knowledge of the model and data can be used to quickly come up with reasonable values for the parameters.

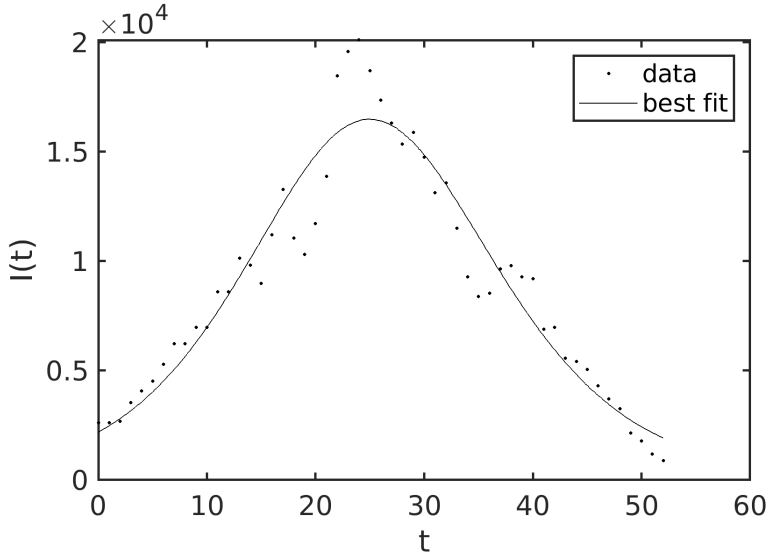


Figure 5.1: Data and best fit for the SIR measles model example

Initial Bounds on Parameters

For $I(0)$, it is reasonable to assume that it will be close to $\bar{I}(0)$. For $S(0)$ we have looked up the population at that time, which turns out to be about 50 million people. This gives us a reasonable upper bound on $S(0)$. We can also get a lower bound by using a simple conservation law,

$$S(0) + I(0) + R(0) = S(T) + I(T) + R(T) \tag{5.15}$$

$$S(0) + I(0) = S(T) + I(T) + \gamma \int_0^T I(t)dt. \tag{5.16}$$

If we make the simplifying assumption that $S(T) = 0$, we have that,

$$S(0) = I(T) - I(0) + \gamma \int_0^T I(t)dt. \tag{5.17}$$

Note, we could have made an assumption of the form $S(T) = \mu S(0)$, $0 \leq \mu < 1$, but this would require some expert knowledge to ensure a reasonable μ be chosen.) For this dataset, our lower bound on $S(0)$ works out to be around 700,000 people. Now, for any given $S(0)$, we can roughly estimate β. This is done by considering $I'(t)$ at $t = 0$,

$$I'(0) = \beta S(0)I(0) - \gamma I(0). \tag{5.18}$$

With our upper and lower bounds on $S(0)$, we find that $\beta \in (2.8 \times 10^{-8}, 2.1 \times 10^{-6})$. Note, here we roughly approximated $I'(0)$ by the divided difference, $\bar{I}(1) - \bar{I}(0)$. Given the data, this was fine, since we could clearly see the upward trend of $\bar{I}(t)$ near zero, but if the data were too noisy, this approach could have given the wrong sign on β. As we will see, these bounds are consistent with the initial guesses generated by Dattner's approach and each of the methods we have proposed.

Application of the two alternative techniques

We now turn our attention to how our alternative techniques for obtaining initial guesses look for this example. First, we have that,

$$\tilde{S}(t) = \frac{I'(t) + \gamma I(t)}{\beta I(t)}. \tag{5.19}$$

For ASME,

$$\begin{aligned} \tilde{S}'(t) &= \frac{I''(t) + \gamma I'(t)}{\beta I(t)} - \frac{I'(t)(I'(t) + \gamma I(t))}{\beta I^2(t)} \\ &= \frac{I''(t)I(t) - [I'(t)]^2}{\beta I^2(t)}. \end{aligned} \tag{5.20}$$

Substituting (5.19) and (5.20) into (5.12),

$$\begin{aligned} \tilde{S}'(t) &= -\beta \tilde{S}(t) I(t) \\ \frac{I''(t)I(t) - [I'(t)]^2}{\beta I^2(t)} &= -\beta I(t) \frac{I'(t) + \gamma I(t)}{\beta I(t)} \\ \frac{I''(t)I(t) - [I'(t)]^2}{I^2(t)} &= -\beta \left(I'(t) + \gamma I(t)\right) \\ \beta &= -\frac{I''(t)I(t) - [I'(t)]^2}{I^2(t)\left(I'(t) + \gamma I(t)\right)}. \end{aligned} \tag{5.21}$$

Thus, we can obtain a linear least squares estimate for β, which can be used in (5.18) to give us an estimate for $S(0)$. For this example, we can obtain reasonable approximations for the required second derivatives. In this numerical experiment, we use forward finite differences to approximate the required derivatives.

For AINT-SME, the integral form of (5.12) is given by,

$$\tilde{S}(t) = S(0) - \int_0^t \beta \tilde{S}(\tau) I(\tau)\, d\tau. \tag{5.22}$$

Substituting in (5.19), we have,

$$\begin{aligned}
\frac{I'(t) + \gamma I(t)}{\beta I(t)} &= S(0) - \int_0^t I'(\tau) + \gamma I(\tau)\, d\tau \\
\frac{I'(t) + \gamma I(t)}{\beta I(t)} &= S(0) - I(t) + I(0) - \gamma \int_0^t I(\tau)\, d\tau \\
\frac{I'(t)}{I(t)} + \gamma &= \beta S(0) + \beta \left(-I(t) + I(0) - \gamma \int_0^t \bar{I}(\tau)\, d\tau \right)
\end{aligned} \tag{5.23}$$

While this expression is not linear in β and $S(0)$, it is linear in $b = [\beta S(0), \beta]$. Once we obtain a linear least squares estimate of b, we can recover β and $S(0)$ from, $S(0) = \frac{b_1}{b_2}$ and $\beta = b_2$. As with ASME, the main advantage of this approach is that it is very simple - only requiring us to approximate $I'(t)$ and $\int_0^t I(\tau)\, d\tau$. We again note that both of these techniques result in linear least squares problems and hence require no nonlinear optimization at all.

Experimental Results

For this example, we consider the cost of the approaches we have discussed for obtaining the initial guesses for the parameters and using *lsqnonlin* in Matlab to obtain the final estimates (Tables 5.1 and 5.2). The Jacobian required by the optimizer is approximated by simultaneously simulating the variational equations along with the state vector. For the Random column, we drew 50 uniform random values for $S(0)$, based on the initial bounds we obtained above. For each value of $S(0)$, we obtain our initial guess for β using (5.18). We report the range of times the optimizations took for these 50 random guesses.

Stage	Dattner[2]	ASME	AINT-SME	Data Driven	(5.7)	Random
Initial Guess	0.52	0.0004	0.056	0.21	0.17	0
Optimization	0.66	0.74	0.53	0.65	0.65	1.24-2.55
Total	1.18	0.74	0.59	0.86	0.82	1.24-2.55

Table 5.1: Timing results in seconds for the approaches described above for the Measles model. The times reported in all cases are obtained in Matlab, running on the same hardware.

[2] Note, the code provided in the supplementary material of Dattner's original paper [13] contained a coding error, so the results presented here are for the corrected code.

We observe that Dattner's approach spends almost as much time generating the initial guess as it does performing the final optimization. While ASME quickly produces an initial guess that results in a better than average initial guess, the extra time spent by AINT-SME is justified here, as it provides us with an excellent initial guess. All of these methods are able to generate initial guesses that are better than the 50 random guesses we tried. Our data driven approach produces a similar initial guess to Dattner's approach, but requires less time to generate the initial guess. Using the (5.7) version of our data driven approach saves a small amount of time in generating the initial guess (about 20%). As reference, the initial guesses generated by each method are shown in Table 5.2.

Parameter	Dattner[2]	ASME	AINT-SME	Data Driven	(5.7)	Final Estimates
$I(0)$	2.550e3	2.610e3	2.610e3	2.345e3	2.281e3	2.1975e3
$S(0)$	4.276e6	2.049e6	3.976e6	4.185e6	4.191e6	4.1100e6
β	3.554e-7	6.832e-7	3.829e-7	3.638e-7	3.632e-7	3.7116e-7

Table 5.2: Initial guesses obtained by each method and the final parameter estimates for the Measles example.

Results for additional years

The dataset contains data for about two decades, with every second year exhibiting a clear peak in the incidence of measles. This gives us 8 additional years to compare these methods on. The results (not shown) are similar to the above. All of the approaches, except ASME, are able to provide initial guesses that are sufficiently accurate to allow the final optimization to succeed. ASME appears to have trouble in some years where the model does not match the data as well, since ASME requires a reasonably accurate approximation to $I''(t)$. In terms of cost, Dattner's approach is still the most expensive, since α contains 7 parameters in its nonlinear optimization, while our approaches have fewer. The relative costs of the other methods are similar.

5.2.2 Influenza model example

Our second example is a simple model of influenza. The model is given by,

$$x_1'(t) = -\theta_1 x_1(t) x_3(t), \tag{5.24}$$
$$x_2'(t) = \theta_1 x_1(t) x_3(t) - \theta_2 x_2(t), \tag{5.25}$$
$$x_3'(t) = \theta_3 x_2(t) - \theta_4 x_3(t) \tag{5.26}$$

where θ_{1-4} and the initial conditions are the parameters to be estimated. The model is simulated from $t = 0$ to $t = 10$. For this example, we consider how ASME and AINT-SME perform. As in [13], we consider two cases - one where only x_1 is unobserved and one where only x_2 is unobserved.

Case of x_1 unobserved

First, we have that,

$$\tilde{x}_1(t) = \frac{x_2'(t) + \theta_2 x_2(t)}{\theta_1 x_3(t)}. \tag{5.27}$$

For ASME,

$$\begin{aligned}\tilde{x}_1'(t) &= \frac{x_2''(t) + \theta_2 x_2'(t)}{\theta_1 x_3(t)} - x_3' \frac{x_2'(t) + \theta_2 x_2(t)}{\theta_1 x^2{}_3(t)} \\ &= \frac{x_3(t)\big(x_2''(t) + \theta_2 x_2'(t)\big) - x_3'\big(x_2'(t) + \theta_2 x_2(t)\big)}{\theta_1 x^2{}_3(t)}.\end{aligned} \tag{5.28}$$

Substituting (5.27) and (5.28) into (5.24),

$$\begin{aligned}\tilde{x}_1'(t) &= -\theta_1 \tilde{x}_1(t) x_3(t) \\ \frac{x_3(t)\big(x_2''(t) + \theta_2 x_2'(t)\big) - x_3'\big(x_2'(t) + \theta_2 x_2(t)\big)}{\theta_1 x^2{}_3(t)} &= -\theta_1 x_3(t) \left(\frac{x_2'(t) + \theta_2 x_2(t)}{\theta_1 x_3(t)}\right) \\ \frac{x_3(t)\big(x_2''(t) + \theta_2 x_2'(t)\big) - x_3'\big(x_2'(t) + \theta_2 x_2(t)\big)}{\theta_1 x^2{}_3(t)} &= -x_2'(t) - \theta_2 x_2(t).\end{aligned}$$

Rearranging and simplifying, this reduces to,

$$x_2''(t) - \frac{x_3'(t) x_2'(t)}{x_3(t)} = -\theta_1 x_3(t) x_2'(t) + \theta_2 \left(\frac{x_2(t) x_3'(t)}{x_3(t)} - x_2'(t)\right) - \theta_1 \theta_2 x_2(t) x_3(t) \tag{5.29}$$

Thus, we can obtain a linear least squares estimate for $b = [\theta_1, \theta_2, \theta_1\theta_2]$, with the nonlinear

constraint that $b_1 b_2 = b_3$. The initial condition, $x_1(0)$ can be estimated from (5.27), with $t = 0$.

For AINT-SME, the integral form of (5.24) is given by,

$$\tilde{x}_1(t) = x_1(0) - \int_0^t \theta_1 \tilde{x}_1(\tau) x_3(\tau)\, d\tau. \tag{5.30}$$

Substituting in (5.27), we have,

$$\begin{aligned}
\frac{x_2'(t) + \theta_2 x_2(t)}{\theta_1 x_3(t)} &= x_1(0) - \int_0^t x_2'(\tau) + \theta_2 x_2(\tau)\, d\tau \\
\frac{x_2'(t) + \theta_2 x_2(t)}{x_3(t)} &= \theta_1 x_1(0) - \theta_1\big(x_2(t) - x_2(0) + \theta_2 \int_0^t x_2(\tau)\, d\tau\big)
\end{aligned}$$

Rearranging and simplifying, this reduces to,

$$\frac{x_2'(t)}{x_3(t)} = \theta_1 x_1(0) - \theta_1\big(x_2(t) - x_2(0)\big) - \theta_1\theta_2 \int_0^t x_2(\tau)\, d\tau - \theta_2 \frac{x_2(t)}{x_3(t)}. \tag{5.31}$$

This results in an equation that is not linear in the original parameters, but is linear in $b = [\theta_1 x_1(0), \theta_1, \theta_1\theta_2, \theta_2]$, with the nonlinear constraint that $b_2 b_4 = b_3$. Once we obtain a linear least squares estimate of b, we can recover our original 3 parameters, since $x_1(0) = \frac{b_1}{b_2}$. Lastly, we note that since θ_3 and θ_4 only appear in (5.26), and we have data for both $x_2(t)$ and $x_3(t)$, we can obtain initial guesses for these two parameters using either SME or INT-SME.

Case of x_2 unobserved

First, we have that,

$$\tilde{x_2}(t) = \frac{x_3'(t) + \theta_4 x_3(t)}{\theta_3}. \tag{5.32}$$

For ASME,

$$\tilde{x}_2'(t) = \frac{x_3''(t) + \theta_4 x_3'(t)}{\theta_3} \tag{5.33}$$

Substituting (5.32) and (5.33) into (5.25),

$$\begin{aligned}
\tilde{x}_2'(t) &= \theta_1 x_1(t) x_3(t) - \theta_2 \tilde{x}_2(t) \\
\frac{x_3''(t) + \theta_4 x_3'(t)}{\theta_3} &= \theta_1 x_1(t) x_3(t) - \frac{\theta_2}{\theta_3}\big(x_3'(t) + \theta_4 x_3(t)\big) \\
\frac{x_3''(t) + \theta_4 x_3'(t)}{\theta_3} &= -x_1'(t) - \frac{\theta_2}{\theta_3}\big(x_3'(t) + \theta_4 x_3(t)\big).
\end{aligned}$$

Note that above we have used the fact that $x_1'(t) = -\theta_1 x_1(t) x_3(t)$. Rearranging and simplifying, this reduces to,

$$x_3''(t) = -\theta_3 x_1'(t) - (\theta_2 + \theta_4) x_3'(t) - \theta_2 \theta_4 x_3(t). \tag{5.34}$$

Thus, we can obtain a linear least squares estimate for $b = [\theta_3, \theta_2 + \theta_4, \theta_2\theta_4]$, with the constraint that $b \geq 0$. Since θ_2 and θ_4 appear identically in this equation, we assume it is known that $\theta_2 \leq \theta_4$ so that we can distinguish between them. With $\theta_2 + \theta_4 = b_2$ and $\theta_2\theta_4 = b_3$, it follows that $\theta_4^2 - b_2\theta_4 + b_3 = 0$. Solving this quadratic in θ_4, we can take θ_4 to be the largest root and θ_2 is then the smallest root.
For AINT-SME, the integral form of (5.25) is given by,

$$\tilde{x}_2(t) = x_2(0) - \int_0^t \theta_1 x_1(\tau) x_3(\tau)\, d\tau - \theta_2 \int_0^t \tilde{x}_2(\tau)\, d\tau. \tag{5.35}$$

Substituting in (5.32) and again using the fact that $x_1'(t) = -\theta_1 x_1(t) x_3(t)$, we have,

$$\begin{aligned}
\frac{x_3'(t) + \theta_4 x_3(t)}{\theta_3} &= x_2(0) - x_1(t) + x_1(0) - \frac{\theta_2}{\theta_3} \int_0^t \big(x_3'(\tau) + \theta_4 x_3(\tau)\big)\, d\tau \\
x_3'(t) + \theta_4 x_3(t) &= \theta_3 x_2(0) - \theta_3\big(x_1(t) - x_1(0)\big) - \theta_2\big(x_3(t) - x_3(0)\big) - \theta_2\theta_4 \int_0^t x_3(\tau)\, d\tau.
\end{aligned}$$

Rearranging and simplifying, this reduces to,

$$x_3'(t) = \theta_3\big(-x_1(t) + x_1(0)\big) + \theta_3 x_2(0) - \theta_2\big(x_3(t) - x_3(0)\big) - \theta_2\theta_4 \int_0^t x_3(\tau)\, d\tau - \theta_4 x_3(t). \tag{5.36}$$

Again, this results in an equation linear in $b = [\theta_3, \theta_3 x_2(0), \theta_2, \theta_2\theta_4, \theta_4]$. As noted by Dattner, the case of x_2 unobserved has some identifiability issues, so one might consider fixing the initial condition on x_2. In this case, the nonlinear constraints for our problem

are $\frac{b_2}{b_1} = x_2(0)$ and $b_3 b_5 = b_4$. We note that θ_1 can be estimated from (5.24), since we have data for both $x_1(t)$ and $x_3(t)$, using either SME or INT-SME.

Experimental Results

In these experiments, to smooth the data and approximate the required derivatives, we have used splinefit [33], with cubic splines and the placement of knots chosen to give a reasonable fit. For varying n_o, we have generated 100 sets of noisy data, with $\sigma = 0.05$, as was done in [13]. For the optimizations with nonlinear constraints, we use *fmincon* in Matlab, with an interior point algorithm.

Our results for the case of x_1 unobserved are shown in Table 5.3. The quality of the estimates we obtain using AINT-SME are similar to those reported by Dattner in [13], although we have a lower bias in θ_3. This difference appears to be due to the quality of the smoothing of the observed states. Dattner used LPE, with a fixed bandwidth of 1 and first order polynomials. For AINT-SME and ASME we used slightly different meshes for our cubic splines used in splinefit and observe different results, especially for the cases with fewer observations. The guesses for θ_{1-2} and $x_1(0)$ generated using ASME are rather poor. ASME appears to systematically underestimate θ_1, which in turn leads to overestimating $x_1(0)$ in (5.27). AINT-SME takes about $0.063s$ to generate an initial guess and ASME takes about $0.033s$, while Dattner's approach takes around $0.48s$. We also note that ASME and AINT-SME both do not require an initial guess to be provided for any of the parameters, while Dattner's approach requires an initial guess for the parameter α defining the unobserved state.

Our results for the case of x_2 unobserved are shown in Table 5.4. For smoothing the data and approximating the required derivatives, we have used splinefit [33], with cubic splines and one knot every 2 time units. The quality of the estimates we obtain using AINT-SME are similar to those reported in the supplementary material of [13]. The guesses for θ_{2-4} generated using ASME are noticeably worse, although still reasonably close to their true values. AINT-SME takes about $0.035s$ to generate an initial guess and ASME takes about $0.011s$, while Dattner's approach takes around $0.39s$. We again note that ASME and AINT-SME both do not require an initial guess to be provided for any of the parameters, while Dattner's approach requires an initial guess for the parameter α defining the unobserved state.

	Value	$n_o = 12$	$n_o = 21$	$n_o = 51$	$n_o = 101$
$x_1(0)$	1	0.483 (0.078)	0.288 (0.055)	0.172 (0.031)	0.116 (0.035)
$x_2(0)$	0.05	0.050 (0.005)	0.045 (0.002)	0.051 (0.001)	0.051 (0.002)
$x_3(0)$	2	0.051 (0.001)	0.052 (0.002)	0.050 (0.004)	0.052 (0.001)
θ_1	0.3	0.113 (0.089)	0.111 (0.101)	0.065 (0.000)	0.036 (0.021)
θ_2	0.3	0.103 (0.038)	0.078 (0.038)	0.056 (0.015)	0.038 (0.019)
θ_3	1	0.337 (0.297)	0.213 (0.117)	0.170 (0.028)	0.160 (0.013)
θ_4	0.5	0.119 (0.107)	0.072 (0.041)	0.059 (0.011)	0.057 (0.005)
$x_1(0)$	1	1.947 (1.475)	1.840 (1.560)	1.849 (1.694)	1.923 (1.802)
$x_2(0)$	0.05	0.047 (0.004)	0.051 (0.002)	0.037 (0.002)	0.031 (0.007)
$x_3(0)$	2	0.053 (0.001)	0.053 (0.001)	0.037 (0.010)	0.035 (0.010)
θ_1	0.3	0.183 (0.166)	0.191 (0.183)	0.196 (0.192)	0.201 (0.199)
θ_2	0.3	0.195 (0.071)	0.163 (0.071)	0.162 (0.114)	0.163 (0.131)
θ_3	1	0.264 (0.051)	0.242 (0.008)	0.149 (0.036)	0.113 (0.032)
θ_4	0.5	0.090 (0.016)	0.082 (0.002)	0.050 (0.014)	0.039 (0.011)

Table 5.3: Summary of results for the Influenza model, with $x_1(t)$ unobserved, using AINT-SME (top) and ASME (bottom). As in [13], we report the mean squared error and bias (in brackets) for each parameter.

	Value	$n_o = 12$	$n_o = 21$	$n_o = 51$	$n_o = 101$
$x_1(0)$	1	0.046 (0.001)	0.047 (0.001)	0.044 (0.009)	0.031 (0.004)
$x_3(0)$	2	0.046 (0.003)	0.045 (0.012)	0.043 (0.011)	0.033 (0.011)
θ_1	0.3	0.037 (0.008)	0.037 (0.002)	0.030 (0.010)	0.021 (0.008)
θ_2	0.3	0.051 (0.017)	0.042 (0.012)	0.031 (0.005)	0.023 (0.006)
θ_3	0.3	0.298 (0.135)	0.246 (0.134)	0.205 (0.099)	0.180 (0.093)
θ_4	0.3	0.080 (0.037)	0.071 (0.039)	0.061 (0.034)	0.052 (0.030)
$x_1(0)$	1	0.051 (0.006)	0.046 (0.005)	0.037 (0.001)	0.033 (0.006)
$x_3(0)$	2	0.050 (0.001)	0.049 (0.000)	0.041 (0.013)	0.031 (0.007)
θ_1	0.3	0.041 (0.003)	0.032 (0.008)	0.025 (0.003)	0.022 (0.008)
θ_2	0.3	0.222 (0.200)	0.187 (0.151)	0.158 (0.122)	0.116 (0.082)
θ_3	1	0.626 (0.288)	0.542 (0.038)	0.317 (0.042)	0.250 (0.049)
θ_4	0.5	0.382 (0.107)	0.338 (0.174)	0.327 (0.283)	0.284 (0.254)

Table 5.4: Summary of results produced using AINT-SME (top) and ASME (bottom) for the case of unobserved x_2 ($x_2(0)$ fixed at 0.05) for the Influenza model. As in [13], we report the mean squared error and bias (in brackets) for each parameter.

5.2.3 CSTR test problem with C_{out} unobserved

This is a test problem considered in [40]. The continuously stirred tank reactor (CSTR) model is given by,

$$C'_{out}(t) = -\beta(t)C_{out}(t) + F_{in}(t)(C_{in}(t) - C_{out}(t)), \tag{5.37}$$

$$T'_{out}(t) = 130\beta(t)C_{out}(t) + F_{in}(t)(T_{in}(t) - T_{out}(t)) + \alpha(t)(T_{cool}(t) - T_{out}(t)), \tag{5.38}$$

$$= 130\beta(t)C_{out}(t) + X(t) \tag{5.39}$$

where the coefficients $\beta(t)$ and $\alpha(t)$ are given by,

$$\beta(t) = \kappa \exp\left(-10^4\tau\left(\frac{1}{T_{out}(t)} - \frac{1}{T_{ref}}\right)\right), \tag{5.40}$$

$$\alpha(t) = \frac{aF_{cool}(t)^{b+1}}{F_{cool}(t) + \frac{aF_{cool}(t)^b}{2}}, \tag{5.41}$$

and T_{ref} is a fixed reference temperature chosen to be 350. The model parameters are κ, τ, a, and b; with values of 0.461, 0.83301, 1.678, and 0.5, respectively. Parameter b is considered known, due to identifiability issues. The initial conditions need to be estimated and their true values are $C_{out}(0) = 1.5965$ and $T_{out}(0) = 341.3754$. These values correspond to a steady state of the system at $t = 0$. The model runs from $t = 0$ to $t = 64$. Every 4 time units, one of the five inputs is stepped up or down, as summarized in Table 5.5. Observations are made every $\frac{1}{3}$ time units, resulting in 192 observation times. Simulated data is generated by adding white noise with standard deviations of 0.0223 for C_{out} and 0.79 for T_{out}. As discussed in [40], this level of noise is typical in many chemical engineering processes. The true trajectories corresponding to these parameters are shown in Figure 5.2.

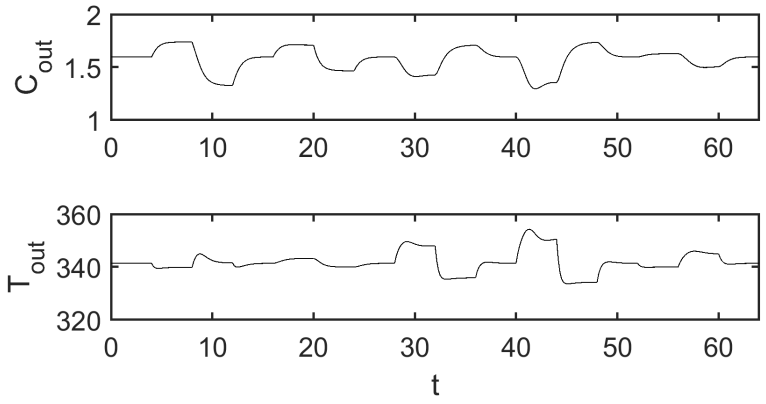


Figure 5.2: True trajectories for the CSTR test problem

In [40], Ramsay reported results for the case where the model is fully observed and the

time	F_{in}	C_{in}	T_{in}	T_{cool}	F_{cool}
0	1	2	323	335	15
4	**1.5**	2	323	335	15
8	**0.5**	2	323	335	15
12	1	2	323	335	15
16	1	**2.2**	323	335	15
20	1	**1.8**	323	335	15
24	1	2	323	335	15
28	1	2	**343**	335	15
32	1	2	**303**	335	15
36	1	2	323	335	15
40	1	2	323	**340**	15
44	1	2	323	**330**	15
48	1	2	323	335	15
52	1	2	323	335	**20**
56	1	2	323	335	**10**
60	1	2	323	335	15

Table 5.5: Summary of how the inputs are stepped up and down every 4 time units in the CSTR test problem.

case where only T_{out} is observed. In this second case, we can apply one of our techniques in order to obtain an initial guess. We have used AINT-SME in this example. Specifically, since (5.39) is linear in $C_{out}(t)$, we can simply rearrange to obtain,

$$\tilde{C}_{out}(t) = \frac{T'_{out}(t) - X(t)}{130\beta(t)}. \tag{5.42}$$

Following the AINT-SME procedure, if we integrate (5.37), we obtain,

$$\begin{aligned}\frac{T'_{out}(t) - X(t)}{130\beta(t)} - C_{out}(0) = \frac{1}{130}\int_0^t X(\tau)\,d\tau - \frac{T_{out}(t) - T_{out}(0)}{130} + \int_0^t F_{in}(\tau)C_{in}(\tau)\,d\tau \\ - \int_0^t \frac{F_{in}(\tau)T'_{out}(\tau)}{130\beta(\tau)}\,d\tau + \int_0^t \frac{F_{in}(\tau)X(\tau)}{130\beta(\tau)}\,d\tau.\end{aligned} \tag{5.43}$$

Unlike our previous examples, this expression can not be formulated as a linear least squares problem with nonlinear constraints. We do note though that κ appears linearly. For this experiment, we generated 100 sets of noisy data as described above. We then used AINT-SME and our data driven approach to obtain initial guesses for the parameters. Since both of theses approaches result in non-linear least squares problems, we use LHS to

obtain a suitable starting point for Matlab's lsqnonlin routine (using the default Trust-Region-Reflective algorithm). In order to apply LHS and fully specify the problem, we have assumed lower bounds on the model parameters, $[\kappa,\ \tau,\ a,\ C(0)]$, to be given by $[0.01, 0.01, 0.1, 0.3]$ and upper bounds of, $[2, 2, 2.5, 3.2]$. We do not specify a bound on $T(0)$, since we have an observation available. The final estimates were then found by running an LM optimizer, with the required sensitivities approximated using FD. Results are presented in Table 5.6.

We see that our initial guesses appear to be quite close to the true parameter values, allowing the final optimization to quickly converge to the true solution in most cases. For AINT-SME, we originally found that some initial guesses were leading to poor local minima being found. To prevent this, we made the code more robust by checking what value of the objective function was obtained with the initial guess. If the objective function was too large (in this example, we choose 1700), then we ran several iterations of fminsearch before performing the final gradient based optimization. This modification resulted in only 2 cases where the initial guesses led to poor local minima, so we exclude these 2 cases from the summary results we report. We left in this modification when we ran our data driven approach and found that it is only occasionally required. As mentioned, this modification makes the code more robust, but it does increase the computer time somewhat. On average, AINT-SME required 0.16s to generate the initial guess and 1.96s to perform the full procedure. Our data driven approach took 0.23s to generate the initial guess and only 0.78s to perform the full procedure. We see that our data driven approach takes a bit longer to generate the initial guess, but the quality of the guess results in the final optimization only taking about 30% as long, compared to when AINT-SME was used to generate the initial guess.

	κ	τ	a	C(0)	T(0)
mean guess	0.44532	0.91377	1.47161	1.83897	339.15659
stdev of guess	0.03491	0.04038	0.12359	0.14134	1.88204
mean guess	0.42926	0.84455	1.49866	1.66196	341.15355
stdev of guess	0.02612	0.03412	0.09714	0.18818	1.73612
mean estimate	0.46003	0.83151	1.67580	1.57860	341.49138
stdev of estimate	0.00879	0.00815	0.03839	0.09148	0.80550

Table 5.6: Estimation results for the CSTR problem with only T_{out} observed, for AINT-SME (top), our data driven approach (middle) and the final estimates (bottom).

5.2.4 Fitz-Hugh Nagumo test problem with $R(t)$ unobserved

We now consider the Fitz-Hugh Nagumo model, with $R(t)$ unobserved. The experimental setup is the same as described in Section 2.6.1, but with $R(t)$ unobserved and the true values of parameters a, b, and c set to 0.2, 0.2, and 3, respectively (as used in [40]). For this example, we have,

$$\tilde{R}(t) = \frac{V'}{c} - V(t) + \frac{V^3(t)}{3}. \tag{5.44}$$

This results in ASME being, after some manipulation,

$$V''(t) + V(t) = a + b\big(V(t) - \frac{V^3(t)}{3}\big) - \frac{b}{c}V'(t) + cV'(t)\big(1 - V^2(t)\big). \tag{5.45}$$

This equation is linear in $\alpha = [a, b, \frac{b}{c}, c]$, with the constraint that $\frac{\alpha_2}{\alpha_4} = \alpha_3$. Note, the initial condition, $R(0)$ can be recovered from (5.44) evaluated at $t = 0$. We also have that AINT-SME is given by,

$$\begin{aligned}\int_0^t V(\tau)\,d\tau + V'(t) = at + b\Big(\int_0^t V(\tau)\,d\tau - \int_0^t \frac{V^3(\tau)}{3}\,d\tau\Big) \\ - \frac{b}{c}\big(V(t) - V(0)\big) \\ + c\big(V(t) - \frac{V^3(t)}{3}\big) \\ + cR(0).\end{aligned} \tag{5.46}$$

This equation is linear in $\alpha = [a, b, \frac{b}{c}, c, cR(0)]$, with the constraint that $\frac{\alpha_2}{\alpha_4} = \alpha_3$.
Numerical results are shown in Figure 5.3. For this experiment, we have added noise with $\sigma = 0.05$ to the true trajectory for 20 sets of simulated data. This is done for two different values of n_o. For ASME, we have used LPE to smooth the data and approximate $V'(t)$ and $V''(t)$. For AINT-SME, we have used central finite differences to approximate $V'(t)$ and have not smoothed the data. For our data driven approach, all parameters appear in α, except $V(0)$, so we still have to provide initial guesses (we have used the initial guesses generated by AINT-SME in this case).
ASME's initial guess for $R(0)$ is quite poor, since it is only using a single observation of (5.44) at $t = 0$. AINT-SME slightly overestimates $R(0)$, while our data driven approach slightly underestimates $R(0)$. ASME provides the worst initial guesses, while the other two approaches provide similarly good initial guesses, although all 3 approaches provide

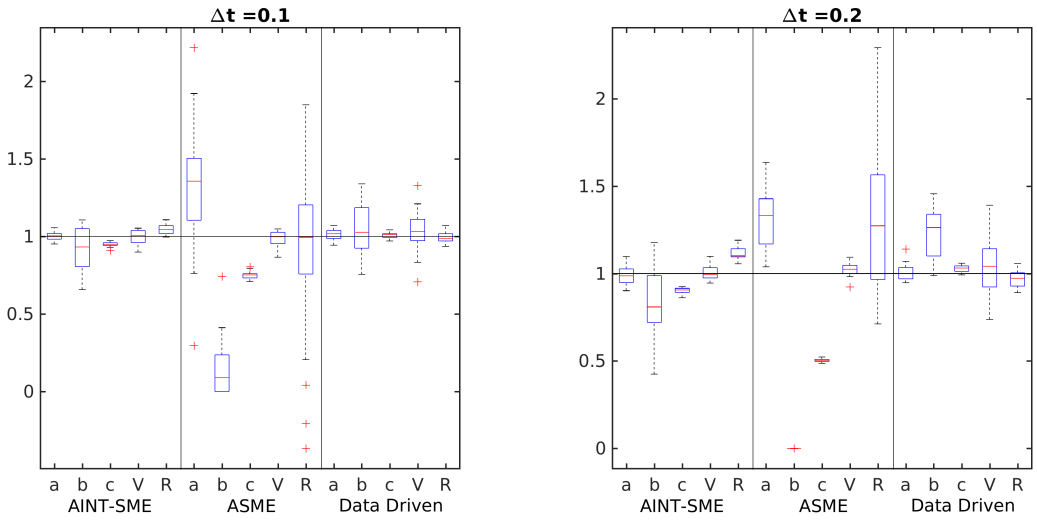


Figure 5.3: Boxplot of initial guesses for the Fitz-Hugh Nagumo model with $R(t)$ unobserved. The left plot is for the case of $\Delta t = 0.1$ ($n_o = 201$) and the right plot is for the case of $\Delta t = 0.2$ ($n_o = 101$).

worse initial guesses as the number of observations decreases, with noticeable biases.

In terms of computer time, the data driven approach takes about 0.147s, AINT-SME takes 0.022s, and ASME takes 0.065s. ASME takes more time than AINT-SME due to the fact that we used LPE to smooth the data in ASME and not in AINT-SME. For our data driven approach, we have included the time taken by AINT-SME, since it was used to generate the initial value for α.

5.2.5 Zebrafish model with $R(t)$ unobserved

This is an example that was also considered in [13], which uses real data that was studied in [22]. The model describes the behaviour of a neuron [53] and is given by,

$$V'(t) = p_1 + p_2 V(t) + p_3 V^2(t) + p_4 V^3(t) + p_5 R(t) + p_6 V(t) R(t), \tag{5.47}$$

$$R'(t) = a_1 + a_2 V(t) + p_7 R(t). \tag{5.48}$$

The parameters to be estimated are $V(0), R(0),$ and p. We fix a to be $[0.7934, 0.0411]$, as was done in [22], due to identifiability issues. Data is only available for $V(t)$ once every time unit, from $t = 0$ to $t = 1000$. Similar to what was done in [22], we impose the constraint that $10 \leq R(t) \leq 60$. This constraint is enforced by the inclusion of an appropriate penalty function. We found this to be necessary to help avoid uninteresting

local minima.

ASME and AINT-SME

From (5.47), we have that,

$$R(t) = \frac{V'(t) - p_1 - p_2 V(t) - p_3 V^2(t) - p_4 V^3(t)}{p_5 + p_6 V(t)}. \tag{5.49}$$

ASME then becomes,

$$\begin{aligned}\big(a_1 + a_2 V(t)\big)\big(p_5 + p_6 V(t)\big) + \frac{p_6 (V'(t))^2}{p_5 + p_6 V(t)} = {} & p_1 V'(t)\big(p_7 + \frac{p_6}{p_5 + p_6 V(t)}\big) \\ & + p_2\big(p_7 V(t) + \frac{p_6 V(t)}{p_5 + p_6 V(t)} - V'(t)\big) \\ & + p_3\big(p_7 V^2(t) + \frac{p_6 V^2(t)}{p_5 + p_6 V(t)} - 2V(t)V'(t)\big) \\ & + p_4\big(p_7 V^3(t) + \frac{p_6 V^3(t)}{p_5 + p_6 V(t)} - 3V^2(t)V'(t)\big), \\ & - V''(t)\end{aligned}$$

and AINT-SME becomes,

$$\begin{aligned}\frac{V'(t)}{p_5 + p_6 V(t)} = {} & R(0) + a_1 t + a_2 \int_0^t V(s)\, ds \\ & - p_7 \int_0^t \frac{-p_1 - p_2 V(s) - p_3 V^2(s) - p_4 V^3(s)}{p_5 + p_6 V(s)}\, ds \\ & + \frac{p_1 + p_2 V(t) + p_3 V^2(t) + p_4 V^3(t)}{p_5 + p_6 V(t)}.\end{aligned} \tag{5.50}$$

In both cases, we are left with equations that are linear in p_{1-4} and nonlinear in p_{5-7}. For our data driven approach, we need only search over values of p_7 to simulate the unobserved state.

Experimental Results

For these results, we have considered the intial conditions fixed at the values of $V(0) = -20.5693$ and $R(0) = 28.1786$ used in [22]. We found that ASME and AINT-SME do not work well for this example and are only able to find poor local minima. An example

of this is shown in the right of Figure 5.4. We see that the initial guess generated by AINT-SME leads to a poor local minimum. Upon investigation, we found the problem arises in the denominator of (5.49). Since we know that the unobserved state should be bounded, we will encounter problems whenever the denominator is near zero. That is, if $-p_5 \approx p_6 V(t)$. Given the data, we have that $V(t) \in (-21.5515, 17.5446)$. If we substitute in the best fit values for p_5 and p_6, we find that the zero of this line is at -21.2335, which is in the range that the observed values of $V(t)$ take. This means that ASME and AINT-SME are unlikely to provide us with initial guesses for the parameters that put us close enough to the best fit parameters for a gradient based optimizer to converge to the global minimum. Our data driven approach avoids this problem, since it simulates $R(t)$ using (5.47).

Parameter	Best Fit	Initial Guess	
		Data Driven	AINT-SME
p_1	5.007e+00	4.953e+00	4.894e+00
p_2	2.860e-01	2.862e-01	2.681e-01
p_3	-5.095e-03	-1.426e-03	-4.406e-03
p_4	-3.748e-04	-1.803e-04	-3.387e-04
p_5	-1.255e-01	-1.278e-01	-1.190e-01
p_6	-5.919e-03	-6.176e-03	-5.068e-03
p_7	-5.737e-03	-5.243e-03	-5.770e-03

Table 5.7: Initial guesses and best fit parameter estimates for the zebrafish example.

The initial guesses and final parameter estimates found using our data driven approach are shown in Table 5.7, along with the initial guess generated using AINT-SME. We notice that the initial guesses obtained using AINT-SME actually look quite good. However, as noted above, the guesses for p_5 and p_6 are such that a gradient based optimizer converges to a poor local minimum. This demonstrates the challenge of fitting IVPs with sharp peaks. Generating the initial guesses took about 0.5s for both methods, while the full optimization took about 36s for our data driven approach and 60s for AINT-SME. The objective function has a value of 6710.46 for the best fit parameters, which is consistent with the parameters reported in [22]. The observed data and trajectories corresponding to the best fit parameters and initial guess parameters generated using our data driven approach are shown in the left of Figure 5.4.

5.2.6 Enzyme Effusion Problem

Our final test problem is a model of enzyme effusion [48, 49, 46]. The model is given by,

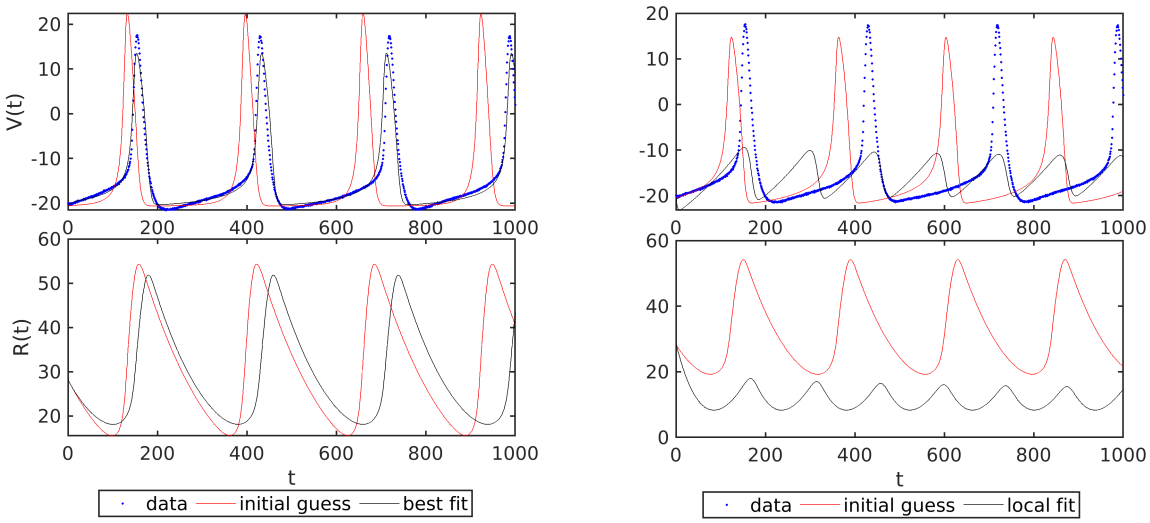


Figure 5.4: Model trajectories for the zebrafish example using our data driven approach (left) and AINT-SME (right) to obtain the initial guess for the parameters.

$$y_1'(t) = p_1(27.8 - y_1(t)) + \frac{p_4}{2.6}(y_2(t) - y(1)) + \frac{4991}{t\sqrt{2\pi}} \exp\big(-0.5(\frac{\ln t - p_2}{p_3})^2\big) \tag{5.51}$$

$$y_2'(t) = \frac{p_4}{2.7}(y_1(t) - y_2(t)) \tag{5.52}$$

where y_1 and y_2 are the intravascular and extravascular enzyme activities, respectively. This model attempts to capture the peak of enzyme activity in the blood that occurs following a heart attack. The initial condition is that both states are initially equal to 27.8, which corresponds to normal enzyme activity, at $t = 0.1$. The other four parameters are to be estimated in this problem. The data for this test problem consists of measurements of y_1. The data are shown in Figure 5.5 and available in tabular form in [48]. Looking at the data, we observe that y_1 does not return to the normal value of 27.8, which it will in the model. This kind of inconsistency between the model and the data may cause ASME or AINT-SME to give poor initial guesses.

For ASME and AINT-SME, we have that the unobserved state can be written as,

$$\tilde{y}_2(t) = \frac{2.6}{p_4}\big(y_1'(t) - p_1(27.8 - y_1(t)) - \frac{4991}{t\sqrt{2\pi}} \exp(-0.5(\frac{\ln t - p_2}{p_3})^2)\big) + y_1(t). \tag{5.53}$$

Applying the ASME procedure and simplifying, we end up with,

$$y_1''(t) = -p_1 y_1'(t) - \frac{p_4}{2.6} y_1'(t) + p_1 p_4 (27.8 - y_1(t)) \\ + \frac{4991}{t^2\sqrt{2\pi}} \left(\frac{p_4}{2.7} t - 1 - \frac{\ln t - p_2}{p_3^2}\right) \exp(-0.5(\frac{\ln t - p_2}{p_3})^2). \tag{5.54}$$

For this example, rather than obtaining an analytical form for AINT-SME, we simply use numerical quadrature to integrate the above ASME formula to approximate the AINT-SME formula.

For our data driven approach, we consider applying it to both INT-SME (5.3) and SME (5.4). We only need an intitial guess for p_4 in order to simulate the unobserved state, but since only p_1 appears linearly in SME and INT-SME, we still need to provide guesses for 3 of the 4 parameters.

To compare how the methods perform on this problem, we look at the shapes of their objective functions (Figure 5.5). To visualize the objective functions, we consider how the minimum of each objective function varies as a function of p_4. We first observe that for the final optimization, we have two solutions with similar objective function values (trajectories and objective function shown in the top of Figure 5.5). The global minimizer has an objective function value of around 63.48. The local minimizer has an objective function value of around 66.23. This second minimum corresponds to the case where p_4 is very large - essentially resulting in $y_1(t) = y_2(t)$ - and is not really of interest.

We observe that ASME contains two local minima, but they are both sufficiently close to the global minimum, although the shape of the objective function is of course dependent on the smoother used when approximating the required derivatives. AINT-SME is only able to find the uninteresting minimum corresponding to large values of p_4. This appears to be due to the data inconsistency we identified above. ASME does not have this problem, since the discrepancy is in the value of the state, not its derivative. When our data driven approach is used to approximate the unobserved state, we observe similar behaviour in the objective functions. The most notable difference is the existence of a minimizer at $p_4 = 0$, which is in the basin of attraction of the global minimizer of interest, for both SME and INT-SME. SME also still exhibits a global minimum close to the solution of interest. While INT-SME with our data driven approach does have a clear global minimizer at $p_4 = 0$, there is still a local minimum corresponding to large values of p_4. We note that AINT-SME and ASME do not have this minimizer at $p_4 = 0$, due to p_4 being in the denominator of (5.53). One might view this case of $p_4 = 0$ as an example of why it is sometimes useful to first fit the parameters of a simpler model, then use those parameters as an initial guess when fitting the more complex model.

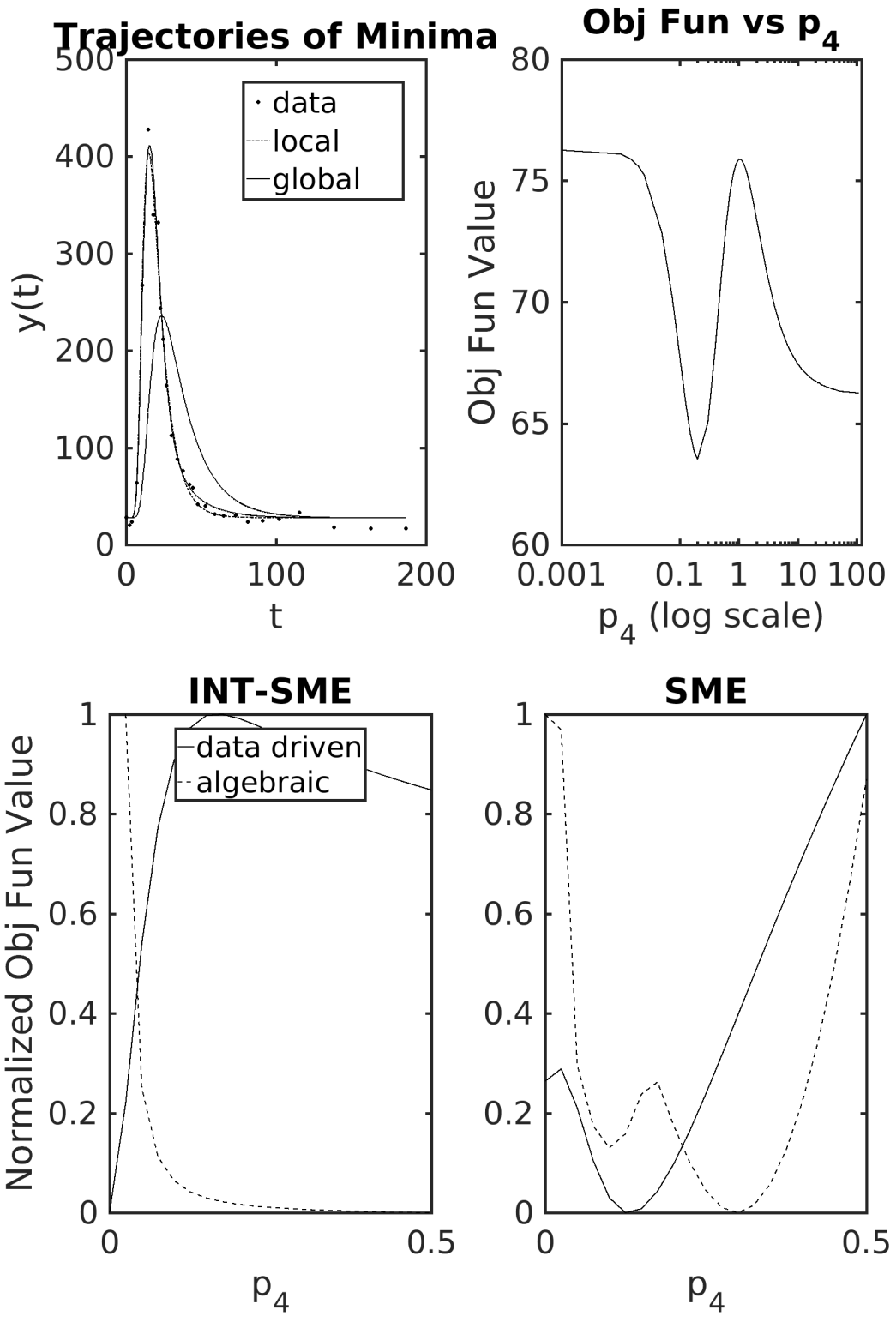


Figure 5.5: Top: Trajectories and Shape of objective function for the Enzyme Effusion Problem. Bottom: Shape of objective functions for initial guess procedures.

Chapter 6

Conclusion

In this thesis, we considered the computationally efficient solution of the parameter estimation problem that arises when fitting ODE and DDE IVPs to observed data. We reviewed several approaches for solving problems of this type and focused our investigations on the single shooting approach, where an effective IVP solver is used to accurately approximate the model trajectory when evaluating the objective function. Using several different approaches from the literature, we demonstrated that it is often the case that a good initial guess p_0 (for the associated NLS problem) can be efficiently obtained by solving a simpler related inverse problem. With a good p_0 available, a gradient based single shooting approach can then often be quite effective for approximating the solution of these parameter estimation problems.

Throughout the thesis we emphasized how the structure of the underlying ODE model could be leveraged. When obtaining p_0, we can make use of the model structure to reduce the cost of approaches like SME and INT-SME by considering subsets of parameters that appear independently in $f(t, y, p)$. Furthermore, we used the linearity of some parameters to reduce the dimensionality of the search space when searching for p_0. To perform the search over the nonlinear parameters, we recommended using LHS, which was previously suggested for use with searching for the minimizer of (1.7). In the case of unobserved components of the state vector, we demonstrated that the structure of the model can often be used to reduce the difficulty introduced by having unobserved states. Specifically, we gave several examples where ASME and AINT-SME resulted in linear least squares problems with non-linear constraints. In these cases, we can obtain initial guesses on the parameters by first ignoring the constraints, then enforcing them. We compared our approaches with a recent technique proposed by Dattner [13], which suffers from having to provide initial guesses for the parameter vector, α, parameterizing the unobserved states.

Several methods for the efficient computation of the model sensitivities required by a Levenberg-Marquardt least squares optimizer were compared and implemented in a parallel computing environment. Numerical experiments suggested that using forward GFM can be the most efficient of these methods in a parallel environment, due to the fact that it can be parallelized across the time domain rather than dividing the work into subsets of parameters, as is naturally done for the other methods. In applications where not much accuracy is required, and there are as many processors available as there are parameters, simply using divided differences is found to be quite effective. If high accuracy is required, we found that simulating the variational equations is recommended.

We also investigated the effectiveness of the adjoint method for both ODEs and DDEs when approximating the gradient of (1.7). We found that the adjoint approach is quite sensitive to the number of observations. In the case of ODEs, we consider using a lower order IVP solver to simulate the adjoint IVP or using parallelism to reduce the cost associated with a large number of observations. For DDEs, this sensitivity is due to the propagation of the jump discontinuities introduced at each observation time.

6.1 Future Work

Applying these approaches to other classes of differential equations is a potential direction for future research. Partial differential equations (PDEs) and stochastic differential equations (SDEs) are two such classes that are widely studied. In the future, more focus will be on how to adapt these methods to problems with a large number of parameters, where some parameters may not be identifiable. The B3 test problem discussed in Section 2.3.1 is one such example from the systems biology literature. Further work is needed to make this method of parameter estimation more generally available to practitioners as an easy to use set of MATLAB routines and C++ classes.

Bibliography

[1] C.T.H. Baker and C.A.H. Paul. Pitfalls in parameter estimation for delay differential equations. *SIAM Journal on Scientific Computing*, 18(1):305–314, 1997.

[2] E. Balsa-Canto, M. Pfeifer, J. Banga, J. Timmer, and C. Fleck. Hybrid optimization method with general switching strategy for parameter estimation. *BMC Systems Biology*, 2(1), 2008.

[3] H. T. Banks and C. Wang. Sensitivity via the complex-step method for delay differential equations with non-smooth initial data. Technical report, North Carolina State University. Center for Research in Scientific Computation, 2016.

[4] Y. Bard. *Nonlinear parameter estimation*. Academic press, 1974.

[5] R. Bellman and R.S. Roth. The use of splines with unknown end points in the identification of systems. *Journal of Mathematical Analysis and Applications*, 34(1):26–33, 1971.

[6] A. A. Berryman. The origins and evolution of predator-prey theory. *Ecology*, 73(5):1530–1535, 1992.

[7] C.H. Bischof, P.D. Hovland, and B. Norris. On the implementation of automatic differentiation tools. *Higher Order Symbol. Comput.*, 3(21):311–331, 2008.

[8] S. Blanes, F. Casas, J.A. Oteo, and J. Ros. The Magnus expansion and some of its applications. *Physics Reports*, 470(56):151 – 238, 2009.

[9] H.G. Bock and K.J Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *Proceedings 9th IFAC World Congress Budapest*, pages 243–247, 1984.

[10] J. Calver. Improving a Cross Entropy approach to parameter estimation for ODEs and DDEs. *MSc. thesis, Department of Computer Science, University of Toronto*, 2014.

[11] Y. Cao, S. Li, L. Petzold, and R. Serban. Adjoint sensitivity analysis for differential algebraic equations: The adjoint DAE system and its numerical solution. *SIAM J. Sci. Comput.*, 3(24):1076–1089, 2003.

[12] S. Chan, JC Light, and J. Lin. Inelastic molecular collisions: Exponential solution of coupled equations for vibration–translation energy transfer. *The Journal of Chemical Physics*, 49(1):86–97, 1968.

[13] I. Dattner. A model-based initial guess for estimating parameters in systems of ordinary differential equations. *Biometrics*, 71(4):1176, 2015.

[14] I. Dattner and S. Gugushvili. Accelerated least squares estimation for systems of ordinary differential equations. *arXiv preprint arXiv:1503.07973*, 2015.

[15] R. P. Dickinson and R. J. Gelinas. Sensitivity analysis of ordinary differential equation systemsa direct method. *Journal of computational physics*, 21(2):123–143, 1976.

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

[17] A. Gábor and J.R Banga. Robust and efficient parameter estimation in dynamic models of biological systems. *BMC systems biology*, 9(1):74, 2015.

[18] D. Gonze and W. Abou-Jaoudé. The Goodwin model: Behind the Hill function. *PLoS ONE*, 8(8):445–466, 2013.

[19] B. Goodwin. Oscillatory behavior in enzymatic control processes. *Advances in Enzyme Regulation*, 3:425–428, 1965.

[20] S. Gugushvili, C.A.J. Klaassen, et al. $\sqrt{n}$-consistent parameter estimation for systems of ordinary differential equations: bypassing numerical integration via smoothing. *Bernoulli*, 18(3):1061–1098, 2012.

[21] G. Hooker. Forcing function diagnostics for nonlinear dynamics. *Biometrics*, 65(3):928–936, 2009.

[22] G. Hooker and L. Biegler. Ipopt and neural dynamics: Tips, tricks and diagnostics. Technical report, Department of Biological Statistics and Computational Biology, Cornell University, 2007.

[23] W. Horbelt, J. Timmer, and H.U. Voss. Parameter estimation in nonlinear delayed feedback systems from noisy data. *Physical Letters A*, 299(5):513–521, 2002.

[24] G. E. Hutchinson. Circular causal systems in ecology. *Annals of the New York Academy of Sciences*, 50(1):221–246, 1948.

[25] J-T. Hwang, E.P. Dougherty, S. Rabitz, and H. Rabitz. The Green's function method of sensitivity analysis in chemical kinetics. *The Journal of Chemical Physics*, 69(11):5180–5191, 1978.

[26] W. Kermack and A. McKendrick. Contributions to the mathematical theory of epidemics, Part I. *Proc. R. Slat. Soc. A115*, pages 700–721, 1927.

[27] O. Kotte and M. Heinemann. A divide-and-conquer approach to analyze underdetermined biochemical models. *Bioinformatics*, 25(4):519–525, 2009.

[28] M.A. Kramer, J.M. Calo, and H. Rabitz. An improved computational method for sensitivity analysis: Green's function method with AIM. *Applied Mathematical Modelling*, 5(6):432–441, 1981.

[29] F.T. Krogh, J.P. Keener, and W.H. Enright. Reducing the number of variational equations in the implementation of multiple shooting. *Numerical Boundary Value ODEs*, pages 121–135, 1985.

[30] U. Kummer, L. F. Olsen, C. J. Dixon, A. K. Green, E. Bornber-Bauer, and G. Baier. Switching from simple to complex oscillations in calcium signaling. *Biophys. J.*, 79(3):1188–1195, 2000.

[31] S.M. Lenz, J.P. Schlöder, and H.G. Bock. Numerical computation of derivatives in systems of delay differential equations. *Mathematics and Computers in Simulation*, 96:124–156, 2014.

[32] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

[33] J. Lundgren. Splinefit v1.13.0.0. MATLAB Central File Exchange, 2017.

[34] B. Macdonald and D. Husmeier. Gradient matching methods for computational inference in mechanistic models for systems biology: a review and comparative analysis. *Frontiers in bioengineering and biotechnology*, 3:180, 2015.

[35] W. Magnus. On the exponential solution of differential equations for a linear operator. *Communications on pure and applied mathematics*, 7(4):649–673, 1954.

[36] T. Maiwald and J. Timmer. Dynamical modeling and multi-experiment fitting with potterswheel. *Bioinformatics*, 24(18):2037–2043, 2008.

[37] J. Martins, P. Sturdza, and J.J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software (TOMS)*, 29(3):245–262, 2003.

[38] C.G. Moles, P. Mendes, and J.R. Banga. Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome research*, 13(11):2467–2474, 2003.

[39] M. Peifer and J. Timmer. Parameter estimation in ordinary differential equations for biochemical processes using the method of multiple shooting. *IET System Biology*, 1(2):78–88, 2007.

[40] 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*, 69(5):741–796, 2007.

[41] A. Raue, C. Kreutz, T. Maiwald, J. Bachmann, M. Schilling, U. Klingmüller, and J. Timmer. Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics*, 25(15):1923–1929, 2009.

[42] A. Raue, M. Schilling, J. Bachmann, A. Matteson, M. Schelke, D. Kaschek, S. Hug, C. Kreutz, B.D. Harms, F.J. Theis, et al. Lessons learned from quantitative dynamical modeling in systems biology. *PloS one*, 8(9):e74335, 2013.

[43] M. Rodriguez-Fernandez, J.A. Egea, and J.R. Banga. Novel metaheuristic for parameter estimation in nonlinear dynamic biological systems. *BMC Bioinformatics*, 7(1):1–18, 2006.

[44] M. Rodriguez-Fernandez, P. Mendes, and J.R. Banga. A hybrid approach for efficient and robust parameter estimation in biochemical pathways. *Biosystems*, 83(2):248–265, 2006.

[45] R.Y. Rubinstein and D.P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning.* Springer-Verlag, New York, 2004.

[46] R. Scitovski and D. Jukić. A method for solving the parameter identification problem for ordinary differential equations of the second order. *Applied Mathematics and Computation*, 74(2-3):273–291, 1996.

[47] J. Timmer, T. Müller, and W. Melzer. Numerical methods to determine calcium release flux from calcium transients in muscle cells. *Biophysical journal*, 74(4):1694–1707, 1998.

[48] B. van Domselaar and P.W. Hemker. Nonlinear parameter estimation in initial value problems. *Stichting Mathematisch Centrum. Numerieke Wiskunde*, (NW 18/75), 1975.

[49] J.M. Varah. A spline least squares method for numerical parameter estimation in differential equations. *SIAM J. of Sci. and Stat. Comput.*, 3(1):28–46, 1982.

[50] A.F. Villaverde, A. Crombach, D. Cicin-Sain, D. Henriques, E. Balsa-Canto, J. Schmid, J. Jaeger, J.R. Banga, J. Saez-Rodriguez, K. Smallbone, et al. Biopredyn-bench: a suite of benchmark problems for dynamic modelling in systems biology. *BMC systems biology*, 9(1):8, 2015.

[51] B. Wang. Parameter estimation for ODEs using a cross-entropy approach. *MSc. thesis, Department of Computer Science, University of Toronto*, 2012.

[52] L. Wang and J. Cao. Estimating parameters in delay differential equation models. *Journal of Agricultural, Biological, and Environmental Statistics*, 17(1):68–83, 2012.

[53] H. R. Wilson. *Spikes, Decisions and Actions:The Dynamical Foundations of Neuroscience*. Oxford University Press, 1999.

[54] H. Wu, H. Xue, and A. Kumar. Numerical discretization-based estimation methods for ordinary differential equation models via penalized spline smoothing with applications in biomedical research. *Biometrics*, 68(2):344–352, 2012.

[55] H. Zivari-Piran. Efficient simulation, accurate sensitivity analysis and reliable parameter estimation for delay differential equations. *PhD thesis, Department of Computer Science, University of Toronto*, 2009.

[56] H. Zivari-Piran and W.H. Enright. Accurate first-order sensitivity analysis for delay differential equations: Part ii: The adjoint approach. *preprint, Department of Computer Science, University of Toronto*, 2009.

[57] H. Zivari-Piran and W.H. Enright. Accurate first-order sensitivity analysis for delay differential equations. *SIAM Journal on Scientific Computing*, 34(5):A2704–A2717, 2012.

Appendix A

Derivation of Adjoint Method

A.1 Adjoint Method for ODE IVPs

In the following, we consider objective functions of the form,

$$G(y,p) = G(y(p)) = \int_0^T g(y(t,p))\,dt, \tag{A.1}$$

where we are restricting ourselves to objective functions that only depend on p through $y(s,p)$. Let $\lambda^T(t)$ be any vector valued function of dimension n_y, defined for $t \in [0,T]$. Now, consider the perturbed objective function,

$$J(p) = G(y(p)) + \int_0^T \lambda^T(t)\big(y'(t,p) - f(t,y(t,p),p)\big)\,dt. \tag{A.2}$$

Note that the term we have added is zero, since $y(t)$ satisfies the ODE, (1.1). Taking the derivative with respect to the parameters, we obtain,

$$\begin{aligned}\frac{\partial J}{\partial p} &= \frac{dG}{dp} + \int_0^T \lambda^T(t)\left(\frac{dy'}{dp}(t) - \frac{\partial}{\partial p}\Big[f(t,y(t,p),p)\Big]\right)dt \\ &= \frac{dG}{dp} + \int_0^T \lambda^T(t)\Big(\frac{\partial y'}{\partial p}(t) - \frac{\partial f}{\partial y}(t)\frac{\partial y}{\partial p}(t) - \frac{\partial f}{\partial p}(t)\Big)\,dt. \end{aligned}\tag{A.3}$$

From (A.1),

$$\begin{aligned}\frac{dG}{dp} &= \frac{\partial}{\partial p}\Big[\int_0^T g(y(t,p))\,dt\Big] \\ &= \int_0^T \frac{\partial g}{\partial y}(t)\frac{\partial y}{\partial p}(t)\,dt,\end{aligned}$$

and therefore, from (A.3),

$$\frac{\partial J}{\partial p} = \int_0^T \left(\frac{\partial g}{\partial y}(t)\frac{\partial y}{\partial p}(t) + \lambda^T(t)\Big(\frac{\partial y'}{\partial p}(t) - \frac{\partial f}{\partial y}(t)\frac{\partial y}{\partial p}(t) - \frac{\partial f}{\partial p}(t)\Big)\right) dt. \tag{A.4}$$

Using integration by parts, we can express the integral term involving $\lambda^T(t)\frac{\partial y'}{\partial p}(t)$ as,

$$\int_0^T \lambda^T(t)\frac{\partial y'}{\partial p}(t)\,dt = \Big(\lambda^T(t)\frac{\partial y}{\partial p}(t)\Big)\Big|_0^T - \int_0^T \dot{\lambda}^T(t)\frac{\partial y}{\partial p}(t)\,dt, \tag{A.5}$$

where $\dot{\lambda}$ denotes $\frac{d\lambda}{dt}$. From (A.5) and (A.4), we obtain, after re-arranging terms,

$$\frac{\partial J}{\partial p} = \int_0^T \Big(\frac{\partial g}{\partial y}(t) - \lambda^T(t)\frac{\partial f}{\partial y}(t) - \dot{\lambda}^T(t)\Big)\frac{\partial y}{\partial p}(t)\,dt - \int_0^T \lambda^T(t)\frac{\partial f}{\partial p}(t)\,dt + \Big(\lambda^T(t)\frac{\partial y}{\partial p}(t)\Big)\Big|_0^T. \tag{A.6}$$

The adjoint system is defined by requiring that $\lambda^T(t)$ be the solution of the IVP:

$$\dot{\lambda}^T(t) = \frac{\partial g}{\partial y}(t) - \lambda^T(t)\frac{\partial f}{\partial y}(t)\,;\lambda^T(T) = 0. \tag{A.7}$$

This choice for $\lambda^T(t)$ eliminates the integral involving $\frac{\partial y}{\partial p}(t)$ in (A.6) and results in the sensitivities being given by,

$$\begin{aligned}\frac{\partial J}{\partial p} &= -\int_0^T \lambda^T(t)\frac{\partial f}{\partial p}(t)\,dt + \Big(\lambda^T(t)\frac{\partial y}{\partial p}(t)\Big)\Big|_0^T \\ &= -\int_0^T \lambda^T(t)\frac{\partial f}{\partial p}(t)\,dt - \lambda^T(0)\frac{\partial y}{\partial p}(0).\end{aligned} \tag{A.8}$$

In some applications, we might not be interested in the sensitivity of G, but rather in the sensitivity of g at time T. In this case, if we take derivatives of (A.7) and (A.8)

with respect to T, we obtain,

$$\dot{\lambda}_T^T(t) = -\lambda_T^T(t)\frac{\partial f}{\partial y}(t)\,;\lambda_T^T(T) = \frac{\partial g}{\partial y}(T), \tag{A.9a}$$

$$\frac{dg}{dp} = -\int_0^T \lambda_T^T(t)\frac{\partial f}{\partial p}(t)\,dt + \lambda_T^T(0)\frac{dy}{dp}(0), \tag{A.9b}$$

Note that the above equations are defined with t varying from T to 0. Defining $x = T-t$, we can return to the standard situation where the independent variable varies from 0 to T.

A.2 Adjoint Method for constant lag DDE IVPs

We consider here the special case of constant lag DDEs with a delay of the form $\alpha = t-\tau$ and constant history function, $y(t) = y_o$, for $t < 0$. For simplicity, we assume there is only one delay in the following derivation, but everything extends in a straightforward way to multiple delays. For convenience, let $\nu = y(t-\tau)$ and let $\lambda^T(t)$ be any vector valued function of dimension n_y, defined for $t \in [0, T+\tau]$. Similar to the ODE case, we define a perturbed objective function,

$$J(p) = G(y(p)) + \int_0^T \lambda^T(t)\big(y'(t,p) - f(t,y(t),y(t-\tau),p)\big)\,dt. \tag{A.10}$$

Taking the derivative with respect to the parameters, we obtain,

$$\begin{aligned}
\frac{\partial J}{\partial p} &= \frac{dG}{dp} + \int_0^T \lambda^T(t)\left(\frac{dy'}{dp}(t) - \frac{\partial}{\partial p}\Big[f(t,y(t,p),y(t-\tau,p)\Big]\right)dt \\
&= \frac{dG}{dp} \\
&\quad + \int_0^T \lambda^T(t)\Big(\frac{\partial y'}{\partial p}(t) - \frac{\partial f}{\partial y}(t)\frac{\partial y}{\partial p}(t) - \frac{\partial f}{\partial \nu}(t)\big(\frac{\partial y}{\partial p}(t-\tau) + y'(t-\tau)\frac{\partial \alpha}{\partial p}(t)\big) - \frac{\partial f}{\partial p}(t)\Big)\,dt \\
&= \int_0^T \left(\frac{\partial g}{\partial y}(t)\frac{\partial y}{\partial p}(t) + \lambda^T(t)\Big(\frac{\partial y'}{\partial p}(t) - \frac{\partial f}{\partial y}(t)\frac{\partial y}{\partial p}(t) - \frac{\partial f}{\partial \nu}(t)\frac{\partial y}{\partial p}(t-\tau)\Big)\right)dt \\
&\quad - \int_0^T \lambda^T(t)\Big(\frac{\partial f}{\partial \nu}(t)y'(t-\tau)\frac{\partial \alpha}{\partial p}(t) + \frac{\partial f}{\partial p}(t)\Big)\,dt.
\end{aligned}$$

In a similar way to the derivation of (A.5), we can re-write the above expression as,

$$\begin{aligned}\frac{\partial J}{\partial p} = & \int_0^T \big(\frac{\partial g}{\partial y}(t) + \lambda^T(t)\frac{\partial f}{\partial y}(t) - \dot{\lambda}^T(t)\big)\frac{\partial y}{\partial p}(t)\,dt - \int_0^T \lambda^T(t)\frac{\partial f}{\partial \nu}(t)\frac{\partial y}{\partial p}(t-\tau)\,dt \\ & - \int_0^T \lambda^T(t)\Big(\frac{\partial f}{\partial \nu}(t)y'(t-\tau)\frac{\partial \alpha}{\partial p}(t) + \frac{\partial f}{\partial p}(t)\Big)\,dt - \Big(\lambda^T(t)\frac{\partial y}{\partial p}(t)\Big)\Big|_0^T.\end{aligned} \tag{A.11}$$

Now, after a change of variables and rewriting the second integral in (A.11) as,

$$\begin{aligned}\int_0^T \lambda^T(t)\frac{\partial f}{\partial \nu}(t)\frac{\partial y}{\partial p}(t-\tau)\,dt &= \int_{-\tau}^{T-\tau} \lambda^T(t+\tau)\frac{\partial f}{\partial \nu}(t+\tau)\frac{\partial y}{\partial p}(t)\,dt \\ &= \int_0^T \lambda^T(t+\tau)\frac{\partial f}{\partial \nu}(t+\tau)\frac{\partial y}{\partial p}(t)\,dt \\ &\quad + \int_{-\tau}^0 \lambda^T(t+\tau)\frac{\partial f}{\partial \nu}(t+\tau)\frac{\partial y}{\partial p}(t)\,dt.\end{aligned}$$

The second equality follows from being able to extend the integral to T by requiring that $\lambda^T(t) = 0$ for $t \geq T$, and splitting the interval of integration. We now see that we can combine the first integral in this expression, with the first integral in (A.11), and after rearranging,

$$\begin{aligned}\frac{\partial J}{\partial p} &= \int_0^T \big(-\dot{\lambda}^T(t) + \frac{\partial g}{\partial y}(t) - \lambda^T(t)\frac{\partial f}{\partial y}(t) - \lambda^T(t+\tau)\frac{\partial f}{\partial \nu}(t+\tau)\big)\frac{\partial y}{\partial p}(t)\,dt \\ &\quad - \int_0^T \lambda^T(t)\Big(\frac{\partial f}{\partial \nu}(t)y'(t-\tau)\frac{\partial \alpha}{\partial p}(t) + \frac{\partial f}{\partial p}(t)\Big)\,dt \\ &\quad + \Big(\lambda^T(t)\frac{\partial y}{\partial p}(t)\Big)\Big|_0^T + \int_{-\tau}^0 \lambda^T(t+\tau)\frac{\partial f}{\partial \nu}(t+\tau)\frac{\partial y}{\partial p}(t)\,dt.\end{aligned}$$

The adjoint system for this constant lag DDE is defined by requiring that $\lambda^T(t)$ be the solution of the IVP,

$$\dot{\lambda}^T(t) = \frac{\partial g}{\partial y}(t) - \lambda^T(t)\frac{\partial f}{\partial y}(t) - \lambda^T(t+\tau)\frac{\partial f}{\partial \nu}(t+\tau)\,;\lambda^T(t) = 0, \text{ for } t \geq T. \tag{A.12}$$

As in the ODE case, this choice for $\lambda^T(t)$ eliminates the integral involving $\frac{\partial y}{\partial p}(t)$ and results in the sensitivities being given by,

$$\begin{aligned}
\frac{\partial J}{\partial p} &= -\int_0^T \lambda^T(t)\Big(\frac{\partial f}{\partial \nu}(t)y'(t-\tau)\frac{\partial \alpha}{\partial p}(t) + \frac{\partial f}{\partial p}(t)\Big)\,dt \\
&\quad +\Big(\lambda^T(t)\frac{\partial y}{\partial p}(t)\Big)\Big|_0^T + \int_{-\tau}^0 \lambda^T(t+\tau)\frac{\partial f}{\partial \nu}(t+\tau)\frac{\partial y}{\partial p}(t)\,dt \\
\frac{\partial J}{\partial p} &= -\int_0^T \lambda^T(t)\Big(\frac{\partial f}{\partial \nu}(t)y'(t-\tau)\frac{\partial \alpha}{\partial p}(t) + \frac{\partial f}{\partial p}(t)\Big)\,dt \\
&\quad -\lambda^T(0)\frac{\partial y}{\partial p}(0) - \int_{-\tau}^0 \lambda^T(t+\tau)\frac{\partial f}{\partial \nu}(t+\tau)\frac{\partial h}{\partial p}(t)\,dt. \qquad \text{(A.13)}
\end{aligned}$$

Note that because the last integral is for $t \leq 0$, we have replaced $\frac{\partial y}{\partial p}(t)$ with $\frac{\partial h}{\partial p}(t)$ (recall that h is the history function).