NONLINEAR PROGRAMMING APPROACHES FOR EFFICIENT LARGE-SCALE PARAMETER ESTIMATION WITH APPLICATIONS IN EPIDEMIOLOGY

A Dissertation

by

DANIEL PAUL WORD

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2013

Major Subject: Chemical Engineering

ABSTRACT

The development of infectious disease models remains important to provide scientists with tools to better understand disease dynamics and develop more effective control strategies. In this work we focus on the estimation of seasonally varying transmission parameters in infectious disease models from real measles case data. We formulate both discrete-time and continuous-time models and discussed the benefits and shortcomings of both types of models. Additionally, this work demonstrates the flexibility inherent in large-scale nonlinear programming techniques and the ability of these techniques to efficiently estimate transmission parameters even in very large-scale problems. This computational efficiency and flexibility opens the door for investigating many alternative model formulations and encourages use of these techniques for estimation of larger, more complex models like those with age-dependent dynamics, more complex compartment models, and spatially distributed data. However, the size of these problems can become excessively large even for these powerful estimation techniques, and parallel estimation strategies must be explored. Two parallel decomposition approaches are presented that exploited scenario based decomposition and decomposition in time. These approaches show promise for certain types of estimation problems.

DEDICATION

To my dad for inspiring my love of learning

ACKNOWLEDGEMENTS

I have been extraordinarily blessed during my time at Texas A&M University and have learned so much in my field of study as well as about life in general. Numerous people have contributed to my life and education, and here I mention only a few.

I thank my family for their love, support, teaching, and wisdom. I especially credit my parents for the sacrifices they made to raise me as they did. Without their profound influence I would most certainly not be where I am today. My sister has always provided a listening ear, but it is her consistent virtue that means the most to me. My grandparents taught me the value of patience, hard work, and love, and without the presence of my dearest aunt I would certainly be less well-adjusted.

I am forever grateful to my advisor, Dr. Carl Laird. Entering graduate school I expected the tutelage in chemical engineering that he has provided, but I did not expect the education about life in general. His passion for research and education is contagious, and his care for students is obvious. He has provided numerous opportunities for me to advance myself professionally as well as personally, and for these I am grateful. At the start of graduate school I was told that advisor selection was the most important choice I would make. At the end of graduate school I can say with confidence that this is true and that Dr. Laird is a better advisor than I had ever hoped to have.

I appreciate my committee members, Dr. M. Sam Mannan, Dr. Juergen Hahn, Dr. Sergiy I. Butenko, and Dr. Zenón Medina-Cetina, for the helpful comments and suggestions they have made. Thank you all for your service. An extra thanks goes to Dr. Mannan for allowing me to serve as a teaching assistant on one of his teaching trips to China. This experience broadened my view of the world considerably.

I cannot thank my fellow graduate students and research group members enough for their friendships. I am especially grateful to Ian Bowling, Mitch Serpas, and Sean Legg for the help they provided me as I transitioned from chemistry into chemical engineering during my first year of graduate school. Sean, Angelica Mann, and I entered the research group at the same time, and I am thankful for the countless hours we have spent learning together.

I am grateful for the financial support of my graduate studies provided in by Sandia National Laboratories and the Office of Advanced Scientific Computing Research within the DOE Office of Science as part of the Applied Mathematics program. I also gratefully acknowledge partial financial support from the National Science Foundation (CAREER Grant CBET 0955205). I would not have attended graduate school if not for the financial support available.

I also thank Dr. Bill Hart, Dr. Jean-Paul Watson, and Dr. John Siirola from Sandia National Laboratories, Dr. David Woodruff from University of California, Davis, and Dr. Johan Åkesson from Lund University for their collaboration. It has been a pleasure working with these individuals.

Finally, I thank my soon-to-be wife, Julie. Her joy, companionship, and love add incredible meaning to my life and inspire me to continue learning, growing, and developing.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

LIST OF TABLES

1. INTRODUCTION*

Despite the many advances in medicine and public health, infectious diseases remain a significant concern, especially in developing countries where inadequate resources, social influences, and environmental factors may prevent effective, sustained results from public health initiatives (Hethcote, 2000; Finkenstädt et al., 2002; Anderson and May, 1991). One goal of public health programs is to control the spread of infectious diseases and minimize the impact of disease on the population through various control measures such as vaccination programs. However, there are several social, environmental, and biological factors affecting the spread of infectious disease, and the observed temporal dynamics are not always well understood.

The development of reliable, mechanistic models for the spread of infectious diseases remains the subject of extensive research and is desirable from two perspectives. First, from a public health perspective, it is clear that reliable models can greatly aid

*Part of this section is reprinted with permission from "Exploiting Modern Computing Architectures for Efficient Large-Scale Nonlinear Programming" by Zhu, Y., Word, D., Siirola, J., and Laird, C.D., 2009. In: R.M. de Brito Alves, C.A.O. do Nascimento and E.C. Biscaia, Jr., Eds., Computer Aided Chemical Engineering, Copyright 2010 by Elsevier.

Part of this section is reprinted with permission from "Estimation of seasonal transmission parameters in childhood infectious disease using a stochastic continuous time model" by Word, D.P., Young, J., Cummings, D., and Laird, C.D., 2010. In: S. Pierucci and G. Buzi Ferraris, Eds., Computer Aided Chemical Engineering, Copyright 2010 by Elsevier.

Part of this section is reprinted with permission from "A Nonlinear Programming Approach for Estimation of Transmission Parameters in Childhood Infectious Disease Using a Continuous Time Model" by Word, D.P., Cummings, D.A.T., Burke, D.S., Iamsirithaworn, S., and Laird, C.D., 2012. Journal of the Royal Society Interface, Copyright 2012 by The Royal Society.

Part of this section is reprinted with permission from "A Progressive Hedging Approach for Parameter Estimation of Stochastic Nonlinear Programs" by Word, D.P., Watson, J.P., Woodruff, D., and Laird, C.D., 2012. Proceedings of PSE2012, Singapore, Copyright 2012 by Elsevier B.V.

Part of this section is reprinted with permission from "Interior-Point Methods for Estimating Seasonal Parameters in Discrete-Time Infectious Disease Models" by Word, D.P., Young, J.K., Cummings, D.A.T., Iamsirithaworn, S., and Laird, C.D., 2013. Submitted to PLOS One.

Part of this section is reprinted with permission from "Efficient Parallel Solution of Large-Scale Nonlinear Dynamic Optimization Problems" by Word, D.P., Kang, J., Akesson, J., and Laird, C.D., 2013. Submited to Computation Optimization and Applications.

in the decision making process. For example, quantitative long-term dynamic models could be used to determine optimal allocation of limited resources, predict outbreak risk, and perform response planning. Reliable models can be used to quantify the effectiveness of previous response tactics and predict the benefit of future planned responses. Second, from a scientific perspective, the identification of a reliable mechanistic model can improve our understanding and identification of important factors affecting infectious disease dynamics. The dynamics of infectious disease spread are not only a function of the disease itself, but also a function of the social and environmental landscape in which the infection spreads. Therefore, to ensure the reliability of these models, they must be able to describe past system behavior, and reliable estimation of disease model parameters and inputs from existing case data is essential. Ideally, these models will not only reliably describe past system behavior but also be capable of effective extrapolation when given changes in the system inputs.

There are two primary goals of this research. The first is the development of high quality infectious disease models and estimation of suitable model parameters so that historic disease dynamics are reliably captured. This goal requires models of appropriate size and complexity to describe disease transmission and the estimation of suitable temporally varying model parameters. To meet this goal, problem sizes often become very large and require efficient estimation approaches. The development and demonstration of efficient estimation approaches is the second goal. These estimation approaches enable the solution of much larger and more difficult estimation problems than what would be possible with less powerful tools.

1.1 Estimation Problems in Epidemiology

Childhood infectious diseases, such as measles and chickenpox, remain a serious public health concern, especially in developing countries, and are commonly used as a test bed for developing disease models and estimation procedures due to the availability of long-term case count data. Probably the most highly studied dataset for measles has been made available electronically by Grenfell (Bjornstad et al., 2002)[†]. This data set contains yearly reported birth records in addition to biweekly reported measles case counts and has several favorable properties, including a high reporting fraction and temporal resolution. Unfortunately, these favorable characteristics are not typical among other datasets. In fact, for most long-term studies, the only available data are disease case counts (incidence) aggregated over time periods that are longer than the serial interval for the disease. The reporting of this case data differs by origin, with locations reporting data at widely varying intervals such as weekly, monthly, or quarterly. The number of cases that are reported can be significantly lower than the actual number of cases, and the level of this underreporting can differ widely over long time horizons, even in the same location (Bobashev et al., 2000). The incidence is almost always under-reported since the data is typically collected passively, and infected individuals may seek medical attention from health care providers that are not obligated to report. Furthermore, the extent of this under-reporting may be difficult to quantify. Changes in public health policies and administration, as well as changing geographic boundaries such as city expansion, can also result in reporting inconsistencies over the full time horizon. Additionally, little information is known about the number and dynamics of susceptibles within the population, therefore this state variable must be estimated along with the unknown parameters. While each of these difficulties may not be present in all epidemiological datasets, they are representative of those encountered in many historical, passively-reported datasets on the incidence of childhood diseases, and they result in significant challenges for effective parameter estimation.

[†]URL: http://www.zoo.cam.ac.uk/zoostaff/grenfell/measles.htm

In addition to the difficulties inherent in the available data, estimation is further complicated by the structure of the disease models themselves, and a number of modeling and parameter estimation approaches have been proposed. There are two fundamental classes of mechanistic models used for the spread of infectious disease. Individual or agent-based modeling approaches have been used extensively and have been proposed for control strategies (Ferguson et al., 2005), however, the large parameter space of these models often overwhelms the data available to specify those parameters. Compartment-based disease models, such as the classic Susceptible-Infected-Recovered (SIR) models, differ significantly from agent-based models.

Compartment-based disease models describe the dynamics of the disease within the population. In these models the population is assumed to be well mixed with individuals placed into various compartments based on their status with respect to the disease. For example, individuals can be classified as being susceptible to the disease (S), infected but not infectious (E), infected and infectious (I), or recovered and immune (R). Additional compartments are included in many models to represent other stages, such as young children who are temporarily immune to the disease because of antibodies obtained from their mother may be represented by a maternal immunity compartment (M). In general, the classification of the model is determined by the flow of the population from one compartment to another. A few examples include SIR, SEIR, MSEIR, SIRS, SEIRS, SI, and SIS (Hethcote, 2000). In this research we focus on the study of measles dynamics using SIR based models.

In contrast to agent-based models, these models have fewer parameters and can be described by sets of differential or discrete-time equations, allowing for efficient, derivative-based estimation from historical case data. Within this framework, model structures can vary dramatically depending upon the selection of incidence and recovery functions and the discretization strategy selected. While the spread of measles is a continuous process, discrete-time models have been formulated in addition to continuous-time models. Considerations can also be made for age or spatial dynamics. Furthermore, when performing parameter estimation, several measures of fit can be proposed, and while measles spread is inherently stochastic, this characteristic is included in some models and ignored in others.

An examination of measles data shows a strong seasonality in the reported cases leading many models to contain a seasonally varying transmission parameter, but the cause for the seasonality is often not entirely known. Identifying correlations between potential system inputs and transmission dynamics is important for understanding factors that affect disease dynamics. This is especially apparent in long-term models where it is not uncommon for social structure and environmental factors to change significantly over the time horizon studied. A better understanding of these factors is important for improving public health policy and aiding public health officials in establishing appropriate control strategies.

In this work we focus on the estimation of seasonally varying transmission parameters in infectious disease models from real measles case data. We formulate both discrete-time and continuous-time models and discuss the benefits and shortcomings of both types of models. Additionally, we investigate seasonality in multiple model parameters, and show that nearly identical seasonality can be estimated from different model parameters. Using data from multiple locations, the seasonality estimated in these parameters shows strong correlation to school term holidays even across very different social settings and holiday schedules. While this result may not explicitly explain the cause of observed seasonality in measles incidence, it lends support to the assertion that school holidays influence the dynamics.

1.2 Efficient Estimation Approach

Due to the many difficulties inherent in estimation problems involving infectious diseases, an ideal estimation procedure should be flexible enough to accommodate the limitations in the available data and the challenges associated with complex nonlinear models. In addition, one seeks methods that are computationally efficient to enable significant exploration of different model structures along with the promise of tackling larger systems.

Advanced nonlinear programming packages provide an excellent framework for efficient estimation of parameters in infectious disease models from long-term case data. Modern mathematical programming languages (e.g., AMPL (Fourer et al., 1993), GAMS (Rosenthal, 2012), Pyomo (Hart et al., 2012), Modelica (Association et al., 2007)) offer convenient and powerful tools to formulate these estimation problems in a flexible framework that allows for exploration of different model structures. Additionally, these tools provide efficient computation of derivative information through automatic differentiation. This allows for their interfacing with powerful numerical solvers that utilize the available derivative information to efficiently solve the formulated problems. Modelers can therefore easily develop numerous models with varying structure and compare estimation results using these models in a timely manner.

Recent advancements in nonlinear programming tools (Gould et al., 2004) allow efficient solution of sparse problems with hundreds of thousands of variables and constraints using standard desktop computing power (Zavala and Biegler, 2006; Zavala et al., 2008; Laird et al., 2005; van Bloemen Waanders et al., 2003), and large-scale nonlinear programming (NLP) has proven to be an effective framework for optimization of operations and profits within the chemical process industries (Scheu and Marquardt, 2011; Diehl et al., 2002; Zavala et al., 2008; Zhu et al., 2010; Tanaka and

Martins, 2011). In this work, we utilize primal-dual interior-point methods based upon the popular open-source solver Ipopt (Wächter and Biegler, 2006) to solve our estimation problems. This has proven to be a powerful tool for optimization of many large systems, and herein we demonstrate its application to infectious disease problems.

The success of these methods has led to the continued development and growth of the mathematical systems describing the optimization problem to improve model rigor and increase the scope of the optimization, and the scale of the NLP problems of interest to both industry and academia can often outstrip the capacity of modern workstations. While significant theoretical advancements have been made in general NLP algorithms, the solution of very large-scale models remains challenging if not impractical (Hartwich and Marquardt, 2010). Concurrently, the advances in computing clock rates and instruction throughput that we once took for granted have slowed dramatically. Computer chip design companies have instead focused on development of parallel computing architectures such as multi-core and hyper-threading devices (Zhu et al., 2009). These new architectures require advanced algorithms to exploit their parallelism, which means that the "free" performance improvements that we have enjoyed as a result of advances in computing hardware will no longer be possible unless we develop algorithms that are capable of utilizing modern computing architectures efficiently. Exacerbating this issue, modern computing architectures are becoming more complex and the parallel computing community is seeing a shift from the use of general purpose CPUs towards specialized processors and heterogeneous architectures. This reflects a significant paradigm shift in the design and implementation of numerical optimization algorithms that can exploit emerging architectures.

Very large problems also arise in infectious disease modeling, and powerful solution approaches are needed. Many techniques exist to estimate parameters for temporal dynamics in single cities (Finkenstädt and Grenfell, 2000; Word et al., 2012; Cauchemez and Ferguson, 2008; Hooker et al., 2011), but spatio-temporal dynamics across multiple cities is also important (Xia et al., 2004; Jandarov et al., 2013). In England and Wales prior to vaccination, measles was endemic in large cities, but in smaller cities disease fadeout occurred (Xia et al., 2004). Reappearance of the disease would then occur only after a case was imported from a surrounding city where measles was endemic. To capture spatio-temporal dynamics, multi-city models must be developed, but these models can become very large requiring more memory and processing power than a single computer can deliver. To solve these problems efficiently, parallel solution approaches are necessary.

This research explores two approaches for efficient parallel optimization. The first approach recognizes equivalences between traditional parameter estimation problems and stochastic programming problems that allows for the formulation and solution of parameter estimation problems using algorithms designed for stochastic programs. Here, the progressive hedging algorithm is used to estimate seasonally varying transmission parameters for 60 cities in England and Wales. This approach decomposes the problem into multiple scenarios where each city contributes data for one scenario. While not a true spatially distributed model, solving this large-scale problem is the first step to development of a solution approach for a true spatio-temporal disease model. The progressive hedging algorithm is easily implemented in parallel, and thus offers a tool that can be used not only to solve large problems faster than with serial approaches, but can also solve problems so large that they could not be stored in the memory of a serial computer.

While there are many advantages to use progressive hedging, including the ease of problem formulation and implementation, one shortcoming is that the convergence of this algorithm can be very slow in some cases since only an averaging scheme is used to drive convergence. Parallel solution approaches making use of derivative information that give guaranteed optimal convergence are also desirable and is the topic of the second parallel optimization approach.

An alternative to the scenario-based decomposition used in progressive hedging is to perform a decomposition based on time. A parallel Schur-complement decomposition algorithm for nonlinear dynamic optimization problems formulated using the simultaneous approach has been developed. This algorithm exploits structure inherent in nonlinear dynamic problems discretized using orthogonal collocation procedures. This structure allows decomposition using a Schur-complement algorithm for parallel solution of the linear systems resulting from an interior-point solution of these optimization problems. When coupled with a parallel interior-point algorithm, efficient solution of real, large-scale optimization problems is possible. While this approach is efficient for problems with many more algebraic than state variables, our research showed that it was not an effective method for our infectious disease formulation. Therefore, this algorithm is demonstrated for the optimal start up of a combined cycle power plant.

1.3 Dissertation Outline

Many years of epidemiology research has been dedicated to modeling and estimation of infectious disease dynamics, and a review of this relevant work is presented in Section 2. This section highlights papers from several decades of research, and lays the foundation from which our modeling and estimation approaches grew. This review explains the premise of compartment based modeling and details research in discrete-time and continuous-time compartment based models where multiple compartments are included. Several estimation approaches are described that focus primarily on the estimation of seasonally varying transmission parameters. Efficiency of these estimation approaches vary widely with some requiring only minutes and others requiring many hours, and results from several of the studies referenced in this section are compared against our results in future sections.

One of the primary goals of our research is the development and demonstration of efficient and powerful approaches to estimate unknown parameters in infectious disease models. The basis of our estimation approach lies in the use of primal-dual interior-point methods for nonlinear programming. While these techniques were not developed during this research, a basic understanding is important to explain how our estimation formulations are suitable for this solution framework. Section 3 gives an overview of the interior-point methods used for all of the estimations described herein and highlights the components of these methods that can be implemented in parallel to build an efficient parallel algorithm.

In Section 4, we show the flexibility inherent in large-scale nonlinear programming techniques and the ability of these techniques to efficiently estimate transmission parameters in multiple disease models using measles case count data. We utilize interior-point methods to efficiently estimate parameters in multiple discrete-time disease models using measles case count data from three cities. These models include multiplicative measurement noise and have seasonality incorporated into multiple model parameters. Our results show that nearly identical seasonality can be estimated from different model parameters, and that the seasonality estimated in these parameters shows strong correlation to school term holidays even across very different social settings and holiday schedules. In all cases, including for time-series data sets of up to 20 years, we were able to perform the estimations in less than 6 seconds. This computational efficiency and flexibility opens the door for investigating many alternative model formulations and encourages use of these techniques for estimation of larger, more complex time-discretized models like those with age-dependent dynamics, more complex compartment models, and spatially distributed data.

Section 5 presents a nonlinear programming approach for estimating unknown states and the seasonal transmission parameter for measles transmission using a continuous-time model with continuous differential equations and both measurement and model noise. Continuous-time formulations offer several advantages over discrete-time formulations for estimation of infectious disease models. Data can be handled in its native form regardless of the reporting interval. This is demonstrated by using biweekly reported data from London and monthly reported data from New York City and Bangkok. Using data in its native form is a significant advantage for diseases with short serial intervals where it would be unreasonable to have data reported at the same interval. The problem is formulated using the simultaneous approach where all state variables are discretized, and the discretized differential equations are included as constraints, giving a large-scale algebraic nonlinear programming problem that is solved using a nonlinear primal-dual interior-point solver. The overall reliability, flexibility, and efficiency is demonstrated in a simulation study and in estimations performed on real case data from three cities.

The discrete-time and continuous-time models outlined in Sections 4 and 5 both fit the reported data well while demonstrating similar seasonality in the transmission parameter. The quality of fit was nearly identical in both formulations despite the presence of an additional parameter on the incidence in the discrete-time model that was not present in the continuous-time model. This shows that the continuous-time formulation is able to reproduce the data as well as the discrete-time formulations. Since the continuous-time formulation can easily accommodate different reporting intervals and recovery rates, it is preferable. The ability to use data with a reporting interval differing from the serial interval of disease is a significant advantage of the continuous-time model especially for diseases with short serial intervals. Furthermore, it is a more natural description of the continuous process and provides a well developed framework for studying alternative model formulations. Despite the differences, both discrete-time and continuous-time formulations show seasonality having a strong correlation with school term holidays.

Our parameter estimation formulations have focused on temporal dynamics of measles for single cities, however there is a strong need to also understand spatio-temporal dynamics across multiple (often many) cities. While many techniques exist for estimations involving single cities, the scale of spatio-temporal multi-city problems becomes prohibitively large for many solution approaches. This use-case is one of many that motivate the development of parallel algorithms.

Traditional parameter estimation problem formulations and two-stage stochastic programs are structurally equivalent, which allows parallel algorithms for solving these stochastic programs to be used to solve parameter estimation problems. In Section 6 we demonstrate the use of the progressive hedging algorithm to estimate seasonal transmission parameters for a continuous-time measles model using data from 20 years and 60 cities in England and Wales. This algorithm is easily parallelizable and shows promise for solving very large-scale problems that are too large to be formulated on a single computer.

Interior-point methods provide a framework for efficient solution of discretized dynamic optimization problems. Section 7 presents a decomposition strategy applicable to DAE constrained optimization problems. Direct transcription is applied to these problems and the resulting nonlinear program is solved using a parallel interior-point algorithm. In our method, we exploit structure in the system resulting from the transcription method in order to preform linear algebra operations on multiple processors. This algorithm shows significant performance improvements over serial approaches in our case studies.

Section 8 concludes this work. This section summarizes the research described in this document and discusses future research directions.

2. PARAMETER ESTIMATION IN EPIDEMIOLOGY*

This section is composed of three parts. The first section contains the literature review describing previous research in epidemiology from which our work formulating discrete-time models was developed. This includes a thorough description of compartment based modeling and various models that have been formulated for the study of measles. This covers many years of research and brief contributions from each work are noted. Some of this literature review overlaps with that described in the next section which focuses on previous research in epidemiology from which our work in continuous-time models arose. Here, we highlight shortcomings inherent in discrete-time formulations that encourage the development of continuous-time models. Many approaches have been used for estimation with continuous models, and we mention a few here. Any redundancy between sections is only intended to elucidate the contributions of past research to each portion of our work. Finally, we mention our new estimation approach for these challenging problems.

2.1 Discrete-Time Models for Measles

The modeling of infectious disease spread has been an increasingly active area of research for the last century. In 1906, Hamer used the mass action assumption, that incidence is proportional to the product of the densities of susceptibles and

*Part of this section is reprinted with permission from "Estimation of seasonal transmission parameters in childhood infectious disease using a stochastic continuous time model" by Word, D.P., Young, J., Cummings, D., and Laird, C.D., 2010. In: S. Pierucci and G. Buzi Ferraris, Eds., Computer Aided Chemical Engineering, Copyright 2010 by Elsevier.

Part of this section is reprinted with permission from "A Nonlinear Programming Approach for Estimation of Transmission Parameters in Childhood Infectious Disease Using a Continuous Time Model" by Word, D.P., Cummings, D.A.T., Burke, D.S., Iamsirithaworn, S., and Laird, C.D., 2012. Journal of the Royal Society Interface, Copyright 2012 by The Royal Society.

Part of this section is reprinted with permission from "Interior-Point Methods for Estimating Seasonal Parameters in Discrete-Time Infectious Disease Models" by Word, D.P., Young, J.K., Cummings, D.A.T., Iamsirithaworn, S., and Laird, C.D., 2013. Submitted to PLOS One.

infectives, in a discrete-time model to better understand measles (Hamer, 1906). In 1911, Ross used a differential equation model to improve understanding of malaria incidence and control (Ross, 1910). Many other deterministic epidemiology models have been developed, and while many have been unable to predict outbreaks (Bailey and Bailey, 1987; Lotka, 1922; Dietz, 1967, 1988), they were able to analyze important characteristics of infection. For example, Kermack and McKendrick discovered in the 1920s that a threshold density of susceptibles must be exceeded for an epidemic outbreak to occur (Kermack and McKendrick, 1927; McKendrick, 1925). Throughout the century, a host of publications incorporating various phenomena in different formulations have presented various approaches to disease modeling (Hethcote, 2000, 1994).

Infectious disease spread is typically described by one of two fundamental classes of mechanistic models. Agent-based or individual modeling approaches, which have been used to propose control strategies (Ferguson et al., 2005), can suffer from a model parameter space that is too large for the available data to successfully specify parameters. Alternatively, compartment-based modeling can be used to describe the population with respect to various stages of disease infection. Compartment-based modeling is the approach used in this work.

In the compartmental framework the population is divided into various compartments based on their status with respect to the disease (Anderson and May, 1991; Hethcote, 2000; Daley and Gani, 1999; Diekmann and Heesterbeek, 2000). In the basic SIR model, the population is assumed to be well-mixed, and individuals are classified as being susceptible to the disease (S), infected with the disease (I), or recovered from the disease and currently immune (R). Mathematical models based on the compartmental framework can be formulated in both continuous-time (resulting in coupled differential equations) and discrete-time (giving rise to a large set of algebraic or transcendental equations). For a discrete-time model, S_t, I_t, and R_t are the current number of susceptible, infected, and recovered individuals at each time interval t. The infection process that defines the number of new cases in a given time interval is described through the incidence function, which usually depends on the present value of the state variables as well as model parameters Θ, which may themselves depend on time. The recovery process is described by a recovery function that is usually dependent on I_t and Θ only.

The classical incidence function for the number of new cases at interval t is typically defined by $\beta I_t S_t / N_t$. Here, β (known as the transmission parameter) is proportional to the number of adequate contacts for the spread of infections, and N_t is the total population at time t. In a structural sense, this is one of the most basic models of infectious disease dynamics, and more complex compartment structures have been studied (Anderson and May, 1991; Hethcote, 2000, 1994). However, significant flexibility in this basic model is still possible through various definitions of the incidence and recovery functions. In particular, by allowing seasonal model parameters, discrete-time models of this basic structure can have a tremendous capability to fit real world case data (Finkenstädt and Grenfell, 2000; Bjornstad et al., 2002; Grenfell et al., 2002).

Based on work by Soper (Soper, 1929), Fine and Clarkson consider an incidence function where the transmission parameter is allowed to vary with time (Fine and Clarkson, 1982). They estimate values of this temporally varying transmission parameter using measles incidence data collected in England and Wales from 1950-1966. Over this time period, the incidence follows a biennial pattern of alternating major and minor epidemics. Remarkably, however, their estimate of the time-varying transmission parameter has a similar pattern and magnitude in both minor and major epidemic years. Furthermore, this pattern is loosely correlated with school holidays.

This strongly supports the assertion of a relatively consistent, underlying seasonal transmission effect related to school terms. They conclude that the observed biennial pattern is a result of the dynamics of susceptible individuals in the population as driven by births and the infection process.

Semi-mechanistic approaches have also been used to describe the infection process. Ellner et al. coupled the mechanistic compartment balances with a phenomenological model (empirically estimated) to describe the incidence function (Ellner et al., 1998). Using the measles dataset from England and Wales, they estimate a general form for the incidence relationship using both feed forward neural networks and semi-nonparametric models. Probing the input-output behavior of their estimated incidence relationship, they suggest the presence of an underlying seasonal effect and that the incidence function should be nonlinear in I_t. Lui et al. conduct a thorough analysis of the equilibrium behavior and stability properties of various continuous-time compartment models with nonlinear incidence rates of the form $\beta I^p(t)S^q(t)$ (Liu et al., 1987). They conclude that, while values of $q \neq 1$ have no "major effects", altering the value of p from unity can have a significant effect on qualitative, long-term behavior.

There is significant work being done to investigate the homogeneity, or lack thereof, in mixing within populations. Cauchemez and Ferguson confirm that assuming nonlinearity in I_t is necessary to not miss key features of epidemics (Cauchemez and Ferguson, 2008). Keeling and Eames explore the implementation of various techniques from network theory into epidemiology theory to provide a more accurate estimation of mixing networks than the typical random-mixing assumption (Keeling and Eames, 2005). In another paper, Keeling specifically examines using metapopulation models to better understand mixing dynamics (Keeling, 1997). Lloyd analyzed an SIR model using a gamma distributed infectious period rather than the normal exponential infectious period (Lloyd, 2001).

Concerning the spread of measles, the results of Fine and Clarkson and others provide strong evidence of an underlying seasonal mechanism that is linked to school terms (Fine and Clarkson, 1982), although Gomes et al. did not reach that conclusion for Portugal (Gomes et al., 1999). However, the Fine and Clarkson model (Fine and Clarkson, 1982) has been criticized (Mollison and Din, 1993), since its long term behavior does not exhibit the observed two-year periodicity, but rather a periodicity close to three years. It is with this backdrop that Finkenstädt and Grenfell introduce the time-series SIR model that can adequately describe the periodicity of the data by using a seasonally varying transmission parameter (Finkenstädt and Grenfell, 2000).

In Section 4 we extend the ideas of Finkenstädt and Grenfell (Finkenstädt and Grenfell, 2000) and present a large-scale, nonlinear programming approach for efficient estimation of time-series SIR models using existing case data. We make use of the time-series SIR model because of its demonstrated ability to reliably represent measles time-series data and the lack of an assumed functional relationship restricting the shape of the seasonal transmission profile. However, unlike in Finkenstädt and Grenfell (2000), our approach is not one-step-ahead, and we do not require the susceptible dynamics and the time-varying reporting fraction as inputs, but instead estimate them simultaneously with the unknown model parameters. The simultaneous estimation of the reporting fraction has been performed previously with a continuous-time SEIR model, where a generalized profiling estimation approach was used that also estimated the susceptible dynamics and the reporting fraction along with the model parameters (Hooker et al., 2011). Here, we use an interior-point method to estimate seasonal parameters for discrete-time models and investigate seasonality in multiple model parameters.

Our estimation problem described in Section 4 is formulated as a large-scale nonlinear programming problem (NLP). The discrete-time model is written over the entire time horizon of the selected data and included in the formulation as constraints. While this approach produces a very large-scale nonlinear programming problem, it can be solved efficiently using modern NLP solvers. Advancements in NLP algorithms, including the introduction of large-scale nonlinear interior-point methods (Wächter and Biegler, 2006; Byrd et al., 2006, 1999; Waltz et al., 2006; Gould et al., 2004) allow efficient solution of increasingly large problems like those presented here. In addition, this approach is very flexible, allowing one to easily formulate new model structures. For example, in this paper we present models with multiplicative noise in the measurements and seasonality in three different model parameters.

The effectiveness of this overall approach is demonstrated in Section 4 with data from communities above the critical community size (where the disease is endemic). Parameters are estimated using available pre-vaccination measles data from the UK (Bjornstad et al., 2002) and New York City (Yorke and London, 1973) from 1944-1963, as well as pre-vaccination measles data collected in Thailand from 1975-1986 (of Epidemiology, 1986). The school term schedule in Thailand differs significantly from the schedule in the other two locations, making it an excellent complimentary dataset for comparing the seasonality of model parameters with school patterns.

2.2 Continuous-Time Models for Measles

It is clear from measles case data that measles incidence follows strong seasonal patterns. As early as 1929, Soper estimated transmission rates from monthly measles case data and proposed that the seasonality was correlated with school terms (Soper, 1929). In 1982, Fine and Clarkson used measles data from the years 1950 to 1965 in England and Wales to estimate a time varying transmission profile (Fine and

Clarkson, 1982). Over this time horizon the data shows a biennial pattern of alternating large and small measles outbreaks, yet the estimated transmission profile remained very similar from year to year. In addition, the estimated transmission profile appeared to be correlated with school holidays.

Reliable estimation from available data can be challenging. Typically, the only disease data available are the case counts (or incidence) of the disease. However, the number of susceptible individuals in the population also has a significant affect on the dynamics of disease spread, and little quantitative data is typically available describing this population. In addition, due to passive collection, the number of reported cases is often significantly lower than the actual number of cases. This can lead to significant underreporting in the data with some data sets reporting fewer than 5% of the actual number of cases. This has led researchers to consider two-stage approaches where the susceptible dynamics and reporting factor are estimated first (Bobashev et al., 2000) and are then treated as known inputs for estimating transmission parameters (Finkenstädt and Grenfell, 2000). In this section, we estimate the degree of underreporting in case counts and the susceptible dynamics themselves simultaneously with the transmission parameters.

Finkenstädt and Grenfell developed a time-series SIR (TSIR) model as a discrete-time model with 26 discretizations per year which is consistent with the serial interval of measles (Finkenstädt and Grenfell, 2000). This model uses a time varying transmission parameter β with yearly periodicity to give 26 unique values for β to describe the time-profile. Although the transmission parameter is assumed to have yearly periodicity, no strong assumption is made regarding the functional form of the parameter. Instead, the parameter profile is treated as an unknown input to be estimated using case count data by finding a one-step-ahead solution to the estimation problem. The estimates from this model using data from England and Wales strengthened the claim that measles transmission is correlated to school holidays.

While discrete-time models like these have proven useful, they suffer from significant drawbacks. These models are discretized over the serial interval of the disease, and the estimation approach requires that the reporting interval of the disease data be an integer fraction of the serial interval. The England and Wales measles data is reported weekly and the serial interval is two weeks making the approach suitable for this data set, but most existing data sets have reporting intervals longer than the serial interval of the disease. Furthermore, the TSIR model includes a parameter α as an exponent on the incidence ($I_{i+1}{=}\beta_i I_i^{\alpha} S_i$). This parameter is difficult to interpret physically, and it has been conjectured that it simply serves to correct for the fact that the model is discrete and not continuous (Xia et al., 2004; Glass et al., 2003).

Continuous models are not only a more natural framework for modeling the continuous nature of disease dynamics, but they also overcome some of the challenges inherent in many discrete-time models. Continuous-time models allow for varying discretization strategies and allows the discretization to differ from the reporting interval of data and the serial interval of the disease. This allows for data to be used in its native form rather than requiring that the data fit a specific reporting interval as is required in the TSIR approach of Finkenstädt and Grenfell (Finkenstädt and Grenfell, 2000). In addition, for this work, no exponent α is included on the incidence term reducing the number of parameters to be estimated by one. Despite this reduction in parameters, these models have been shown to capture the observed disease dynamics comparably (Cauchemez and Ferguson, 2008; Word et al., 2010).

Several continuous-time disease models have been developed. Greenhalgh and Moneim examined the stability properties of different transmission forms and used simulations to show that different periodic solutions are possible with different types of seasonally varying transmission rates (Greenhalgh and Moneim, 2003). Schenzle developed an age and time dependent differential equation model and used simulations to demonstrate that an age-dependent transmission rate more accurately reproduced actual measles data (Schenzle, 1984). Cauchemez and Ferguson developed a stochastic continuous-time SIR model and used it to estimate transmission profiles from London measles data (Cauchemez and Ferguson, 2008). Cintrón-Arias et al. studied parameter subset selection using a continuous-time SEIRS model with a sinusoidally periodic transmission parameter (Cintrón-Arias et al., 2009). This work explored parameter identifiability using synthetic data.

Another challenge of disease modeling is that the transmission of infectious disease is an inherently stochastic process. The use of deterministic models has proven reasonable for use in large cities above a critical community size, but in smaller cities fadeout is seen and deterministic models perform poorly. For measles, the critical community size has been estimated to be around 300,000 people (Keeling, 1997; Bartlett, 1957). When fadeout is observed, reappearance of the disease is caused by an influx of an infected individual into the susceptible population. Finkenstädt et al. (Finkenstädt et al., 2002) modified their TSIR model to allow for stochasticity, and used Monte Carlo simulations to study the affect of the latent stochastic variability of influx.

Other models have been developed to allow for stochasticity using various Monte Carlo techniques. While useful, Monte Carlo techniques suffer from high cost of computation making them unreasonable for large scale models. For example, Cauchemez and Ferguson presented a stochastic continuous-time model using a statistical approach to analyze time-series epidemic data (Cauchemez and Ferguson, 2008). This approach used a data augmentation method to overcome difficulties in inference and presented a diffusion process that mimics the epidemic process. These systems contain stochasticity in the model along with unmeasured states. Therefore, Cauchemez and Ferguson construct the likelihood of the parameters conditional on the data including an integration over the unknown state space. This approach provides guarantees against bias in the estimated parameters, however, the Metropolis-Hastings MCMC sampling is very computationally expensive requiring 20 hours per run (Cauchemez and Ferguson, 2008). Hooker et al. (Hooker et al., 2011) presented an SEIR model to perform parameter estimation with measles data from Ontario using generalized profiling (Ramsay et al., 2007). This approach estimates state variable trajectories and model parameters using a sequential numerical optimization approach that is much more efficient than MCMC techniques, but this approach can still require about 2 hours per estimation (Hooker et al., 2011), although part of this time was due to the iterative process of setting the smoothing parameter in the problem formulation.

He et al. (He et al., 2010) demonstrated their plug-and-play method as a framework for modeling and inference by performing estimates using weekly measles case count data from the 10 largest cities and 10 small cities from England and Wales. This work used an SEIR model with a seasonally varying transmission parameter. The seasonality of the transmission parameters was fixed to correspond with school term holidays, but the amplitude of the seasonality was estimated. Additionally, the durations of the latent and infectious periods were estimated. This approach required approximately 5 hours per estimation, but reductions in the time requirements could be made by tailoring the procedure specifically for a given problem.

2.3 Summary

This literature is the basis upon which our estimation formulations are derived. While similarities certainly exist between our measles model formulations and those found in previous work, our estimation approach described in Section 3 has not been applied before to these epidemiology problems, and significant advantages are available using our nonlinear programming approach. For example, the time required for estimation using the procedures mentioned above was often several hours, while we demonstrate the ability of our approach to solve similar problems in only a few seconds.

Section 3 describes the estimation framework and tools employed in our work. The specific estimation formulations and results for discrete-time and continuous-time models using measles data are detailed in Sections 4 and 5.

3. NLP FRAMEWORK FOR PARAMETER ESTIMATION*

In this section we describe the nonlinear programming framework we employ to formulate and solve parameter estimation problems. Several modeling tools and languages have been used successfully in our research, and we briefly describe each in Section 3.1.

Primal-dual interior-point methods are utilized for all of our numerical solutions, and we describe the basis of interior-point algorithms in Section 3.2. The interior-point algorithms used in our research have been described previously in the literature and were not developed in our work. The highly successful IPOPT code was used for solution of all measles applications included in this work.

Our research in infectious disease modeling includes discrete-time and continuous-time models. The continuous-time models require discretization to formulate in a nonlinear programming framework. Many discretization techniques can be used, and we outline two strategies for collocation on finite elements in Section 3.3.

3.1 Modeling Tools

Modern modeling languages allow for rapid creation of complex optimization problems and reduce the burden of model development, optimization problem formulation, and interfacing with numerical solvers. Three modeling languages have been used in the work presented here and are briefly described below.

*Part of this section is reprinted with permission from "A Nonlinear Programming Approach for Estimation of Transmission Parameters in Childhood Infectious Disease Using a Continuous Time Model" by Word, D.P., Cummings, D.A.T., Burke, D.S., Iamsirithaworn, S., and Laird, C.D., 2012. Journal of the Royal Society Interface, Copyright 2012 by The Royal Society.

Part of this section is reprinted with permission from "A Progressive Hedging Approach for Parameter Estimation of Stochastic Nonlinear Programs" by Word, D.P., Watson, J.P., Woodruff, D., and Laird, C.D., 2012. Proceedings of PSE2012, Singapore, Copyright 2012 by Elsevier B.V.

Part of this section is reprinted with permission from "Efficient Parallel Solution of Large-Scale Nonlinear Dynamic Optimization Problems" by Word, D.P., Kang, J., Akesson, J., and Laird, C.D., 2013. Submited to Computation Optimization and Applications.

The modeling language used to formulate the estimation problems discussed in Sections 4 and 5 is AMPL (Fourer et al., 1993). AMPL is an algebraic modeling language for optimization problems that provides first and second order derivatives using the AMPL Solver Library (ASL) for automatic differentiation. This package was developed at Bell Laboratories and has been available for a number of years. While the MPL language compiler is commercial, the ASL package is open-source. The efficiency of ASL in calculating derivatives has made it a popular interfacing tool for many numerical solvers.

The estimation problem described in Section 6 was formulated in the Python Optimization Modeling Objects (Pyomo) software package (Hart et al., 2011, 2012). Pyomo provides an optimization and modeling platform in a Python environment (Foundation, 1990). Pyomo is part of the open-source Coopr (COmmon Optimization Python Repository) software library released by Sandia National Laboratories (COOPR, 2008). Pyomo allows users to formulate linear, nonlinear, mixed-integer linear, and mixed-integer nonlinear models and interfaces with many commercial and open-source solvers. Due to the scripting capabilities inherently available through the use of a Python environment, many extensions have been packaged with Coopr that provide additional features. For example tools have been developed to allow the straightforward description of differential equations with automatic discretization through collocation methods, and a disjunctive programming extension can automatically convert disjuncts into Big-M and convex hull formulations. Additionally, tools can be built into Coopr that utilize the Pyomo modeling framework. The Python Stochastic Programming (PySP) package is one example that converts a Pyomo model and data sets into stochastic programming problems (Watson et al., 2012; Hart et al., 2012). Exact Hessian and Jacobian information is provided for Pyomo through the ASL interface.

In Section 7 we formulate optimization problems using the Modelica/Optimica and Python-based modeling framework JModelica.org (Association et al., 2007; Åkesson et al., 2010). This is a comprehensive modeling and optimization tool for large-scale dynamic optimization problems that allows straightforward declaration of dynamic equations and provides automatic discretization of these equations using direct collocation methods. The platform employs compiler technology, symbolic manipulation, and code generation to transform high-level Modelica/Optimica descriptions into efficient executables suitable for linking with numerical solvers. Additionally, we interface with the open-source symbolic framework for automatic differentiation and optimal control, CasADi (Andersson et al., 2012), for efficient automatic differentiation.

One attribute that all of these tools have in common is the ability to provide efficient first and second order derivative information through automatic differentiation. This capability makes it possible to use numerical solvers that require this information without the difficult task of manually specifying derivatives.

3.2 Interior-point Algorithm Description

Ipopt was used directly to solve the estimation problems presented in Sections 4 and 5, and was used as the subproblem solver for the solution approach presented in Section 6. Ipopt also served as the basis for the parallel interior-point algorithm developed in Section 7. Ipopt is an open source primal-dual log-barrier interior-point algorithm for solving large-scale nonlinear (and non-convex) programming problems with inequality constraints, and is available through the COIN-OR foundation (Wächter and Biegler, 2006; Wächter, 2002).

This algorithm considers problems of the form,

$$\begin{aligned} \min_{x \in \Re^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0 \\ & d^L \leq d(x) \leq d^U \\ & x^L \leq x \leq x^U \end{aligned} \tag{3.1}$$

where $f(x)$ and $c(x)$ are assumed to be twice differentiable, d^L and d^U are lower and upper bounds on a general function d, and x_L and x_U are lower and upper variable bounds on x. For ease of notation, the algorithm is described for the following problem formulation,

$$\begin{aligned} \min_{x \in \Re^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0 \\ & x \geq 0. \end{aligned} \tag{3.2}$$

Note that general inequalities can be mapped to equality constraints and simple variable bounds through the addition of slack variables.

A significant challenge in the solution of these problems is identifying the active and inactive sets of variables (i.e. the set of variable bounds that are satisfied with equality at the solution versus the variables that are inside their bounds at the solution). With interior point methods, the inequality constraints are moved into the objective function using a log-barrier term to form the barrier subproblem,

$$\begin{aligned} \min_{x \in \Re^n} \quad & f(x) - \mu \sum_{i=1}^{n_x} \ln(x_i) \\ \text{s.t.} \quad & c(x) = 0. \end{aligned} \tag{3.3}$$

This barrier subproblem is solved (approximately) for a sequence of barrier parameters. It can be shown, under mild conditions, that the sequence of solutions of the barrier subproblem converges to the solution of the original problem. From the first-order optimality conditions for (3.3) the following equations can be derived (Wächter and Biegler, 2006),

$$\begin{aligned} &\nabla f(x) - \nabla c(x)\lambda - \nu = 0 \\ &c(x) = 0 \\ &XVe - \mu e = 0 \end{aligned} \tag{3.4}$$

where $\nabla f(x)$ is the gradient of the objective function, $\nabla c(x)$ is the transpose of the constraint Jacobian, λ and ν are the Lagrange multipliers for the equality constraints and inequalities respectively, X is the diagonal matrix of x_i's, V is the diagonal matrix of ν_i's, and e is a vector of ones. A variant of Newton's method is used to solve equations (3.4) (with modifications to ensure the directions are descent directions). Calculating the step $[\Delta x^T \ \Delta\lambda^T \ \Delta\nu^T]$ requires the solution of the following linear system at each iteration.

$$\begin{bmatrix} H & \nabla c(x) & -I \\ \nabla c(x)^T & 0 & 0 \\ V & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta\nu \end{bmatrix} = - \begin{bmatrix} \nabla f(x) - \nabla c(x)\lambda - \nu \\ c(x) \\ XVe - \mu e \end{bmatrix} \tag{3.5}$$

where H is the Hessian of the Lagrange function and I is the identity matrix. This linear system is solved by first symmetrizing (3.5) and solving the so-called augmented system. Exact Hessian and Jacobian information is typically available through modern modeling packages. A filter-based line-search strategy is utilized to ensure global convergence. Further details concerning Ipopt can be found in the literature (Wächter and Biegler, 2006). This approach has been used effectively on numerous large-scale nonlinear optimization problems (Wächter and Biegler, 2006; Zhu et al., 2010, 2011a,b).

The two most computationally expensive steps at each iteration of this algorithm are the evaluation of the NLP residuals and gradients and solving the linear system (3.5) for primal and dual variable steps. To reduce the time required for these steps, the interior-point code could be implemented in parallel so that the NLP evaluations could be performed in parallel, and some parallel method could be used to solve the linear KKT system. These are the steps taken in Section 7 to develop a parallel algorithm.

3.3 Collocation Approaches

For the discrete-time model formulations presented in Section 4, the model equations are included as equality constraints in the nonlinear programming problem. In this approach, the values of the system states are converged simultaneously with the model parameters. This technique has the potential to be very efficient since the forward problem is converged only once along with the estimation problem. For the continuous-time model formulations presented in Section 5, the process is a bit more complex.

There are two general approaches for the solution of large-scale dynamic parameter estimation problems like that considered for our continuous-time model. The sequential approach considers only the degrees of freedom as optimization variables. This includes the initial conditions for state variables, as well as a discretized time profile for the unknown parameters. A complete simulation of the forward problem is performed at each iteration of the optimization. To make use of modern gradient-based methods, derivative information must also be calculated along the entire time-series simulation. These derivatives can be expensive to calculate, especially for problems with many degrees of freedom. Furthermore, these derivatives can be noisy unless care is taken to ensure consistency of the integrator between runs (Betts and Kolmanovsky, 2002). Noise in the evaluation of sensitivities through the integrator can make these problems very challenging for the optimization solver. The simultaneous approach can be used to overcome these difficulties. In the simultaneous approach, all variables, including the states and the parameters, are discretized and treated as optimization variables. The entire discretized model is included as algebraic constraints in the optimization problem. The optimization problem resulting from the simultaneous approach can be larger than the sequential approach. However, this approach can be significantly faster than the sequential approach since the differential equation model is not converged at every iteration of the optimizer, but rather it is converged simultaneously as constraints to the optimization. Furthermore, accurate derivative information is easily obtained using modern automatic differentiation tools coupled with existing modeling frameworks. Recent advancements in nonlinear programming tools (Gould et al., 2004) allow efficient solution of sparse problems with hundreds of thousands of variables and constraints using standard desktop computing power (Zavala and Biegler, 2006; Zavala et al., 2008; Laird et al., 2005; van Bloemen Waanders et al., 2003). In addition to potential efficiency gains, this simultaneous approach allows intuitive specification of additional constraints on the parameters and the state variables, including restrictions on the form of time-varying parameters. The flexibility of general nonlinear formulations, coupled with the efficiency of large-scale algorithms make the simultaneous discretization approach coupled with general nonlinear programming tools an appropriate framework for efficient parameter estimation in nonlinear infectious disease models.

In this work, we use the simultaneous approach. We use collocation on finite elements to discretize the states into finite elements with fixed stepsize across the entire time horizon (Zavala et al., 2008). This converts the continuous differential equation model into an algebraic model that can be formulated as a nonlinear programming problem. Two collocation techniques are used in this work to discretize the dynamics within these finite elements which then allows the discretized equations to be included as equality constraints in the optimization problem.

In Section 5 we used a fifth-degree Gauss-Lobatto collocation technique (Herman and Conway, 1996). We illustrate this approach by showing the discretization of

$$\frac{dQ}{dt} = f(t) \tag{3.6}$$

within a single finite element i. Let $t_{i,j}$ be the time associated with finite element i and collocation point j. With the 5th degree Gauss-Lobatto strategy, there are 5 collocation points for each finite element i ($t_{i,0}$ through $t_{i,4}$). The times $t_{i,0}$ through $t_{i,4}$ correspond to the locations of the collocation points within the finite element, where $t_{i,2}$ is the central collocation point located in the center of the finite element. The time at $t_{i,1}$ is equal to $t_{i,2} - \sqrt{3/7}\frac{1}{2}\Delta t_i$, and the time at t_3 is equal to $t_2 + \sqrt{3/7}\frac{1}{2}\Delta t_i$, where Δt_i is the length of finite element i. Letting $Q_{i,j}$ be the value of $Q(t_{i,j})$ and letting $f_{i,j}$ be the value of $\frac{dQ}{dt}|_{t_{i,j}}$ (Equation 3.6), the collocation equations become,

$$\begin{aligned} Q_{i,1} = \frac{1}{686} \Big\{ &\left(39\sqrt{21} + 231\right) Q_{i,0} + 224 Q_{i,2} + \left(-39\sqrt{21} + 231\right) Q_{i,4} \\ &+ \Delta t_i \left[\left(3\sqrt{21} + 21\right) f_{i,0} - 16\sqrt{21} f_{i,2} + \left(3\sqrt{21} - 21\right) f_{i,4} \right] \Big\} \end{aligned} \tag{3.7}$$

$$\begin{aligned} Q_{i,3} = \frac{1}{686} \Big\{ & \left(-39\sqrt{21}+231\right) Q_{i,0} + 224 Q_{i,2} + \left(39\sqrt{21}+231\right) Q_{i,4} \\ & + \Delta t_i \left[\left(-3\sqrt{21}+21\right) f_{i,0} + 16\sqrt{21} f_{i,2} + \left(-3\sqrt{21}-21\right) f_{i,4} \right] \Big\} \end{aligned} \tag{3.8}$$

$$\begin{aligned} 0 = \frac{1}{360} \Big\{ & \left(32\sqrt{21}+180\right) Q_{i,0} - 64\sqrt{21} Q_{i,2} + \left(32\sqrt{21}-180\right) Q_{i,4} \\ & + \Delta t_i \left[\left(9+\sqrt{21}\right) f_{i,0} + 98 f_{i,1} + 64 f_{i,2} + \left(9-\sqrt{21}\right) f_{i,4} \right] \Big\} \end{aligned} \tag{3.9}$$

$$\begin{aligned} 0 = \frac{1}{360} \Big\{ & \left(-32\sqrt{21}+180\right) Q_{i,0} + 64\sqrt{21} Q_{i,2} + \left(-32\sqrt{21}-180\right) Q_{i,4} \\ & + \Delta t_i \left[\left(9-\sqrt{21}\right) f_{i,0} + 98 f_{i,3} + 64 f_{i,2} + \left(9+\sqrt{21}\right) f_{i,4} \right] \Big\} \end{aligned} \tag{3.10}$$

In our formulations in Section 5, we include model noise between these finite elements so that, while inside a finite element Equations 3.7 - 3.10 are exact, between finite elements there can be state discontinuities. With model noise between finite elements, there is little benefit in the use of a higher order method. However, we initialize our problem using the deterministic case where no model noise is present (model noise terms are fixed to zero), and for this case this discretization is a high-order method.

In Sections 6 and 7 we use a simultaneous transcription method based on finite elements, with Radau collocation points. See Biegler (2010) for a recent monograph. Lagrange polynomials are used to approximate the state, algebraic, and control input profiles.

The optimization mesh for a differential algebraic equation (DAE) model of the form

$$F(\dot{x}, x, y, u) = 0,\ x(t_0) = x_0 \tag{3.11}$$

is defined by n_e finite elements, with normalized lengths h_i, i=1, ..., n_e, where $\sum_{i=1}^{n_e} h_i$=1. The beginning of each finite element is then given by

$$t_i = t_0 + (t_f - t_0) \sum_{k=1}^{i-1} h_k, \quad i = 1, ..., n_e, \tag{3.12}$$

and the collocation point times are given by

$$t_{i,j} = t_0 + (t_f - t_0) \left(\sum_{k=1}^{i-1} h_k + \tau_j h_i \right), \quad i = 1, ..., n_e, \quad j = 1, ..., n_c, \tag{3.13}$$

where $\tau_j \in (0, 1]$, j=1, ..., n_c are the Radau collocation points.

At each collocation point, the discretized variable vectors, $\dot{x}_{i,j}$, $x_{i,j}$, $y_{i,j}$, and $u_{i,j}$ for i=1, ..., n_e and j=1, ..., n_c, are introduced. In addition, state variables at the beginning of each finite element, $x_{i,0}$ for i=1, ..., n_e, are introduced. In each finite element, the differentiated variables are approximated by

$$x(t) = \sum_{k=0}^{n_c} x_{ik} L_k^{n_c+1} \left(\frac{t - t_i}{h_i(t_f - t_0)} \right), \ t \in [t_i, t_{i+1}] \tag{3.14}$$

where $L_j^{n_c+1}(\tau)$ are Lagrange polynomials of order n_c which are computed based on the points $\tau_0, ..., \tau_{n_c}$, with τ_0=0. Accordingly, the expressions for the derivatives $\dot{x}_{i,j}$ are given by

$$\dot{x}_{i,j} = \frac{1}{h_i(t_f - t_0)} \sum_{k=0}^{n_c} x_{i,k} \dot{L}_k^{n_c+1}(\tau_j), \ i = 1, ..., n_e, \ j = 1, ..., n_c. \tag{3.15}$$

At each collocation point, $t_{i,j}$, the DAE relation (3.11)

$$F(\dot{x}_{i,j}, x_{i,j}, y_{i,j}, u_{i,j}) = 0, \ i = 1, ..., n_e, \ j = 1, ..., n_c \tag{3.16}$$

holds. In addition, continuity constraints for the state variable profiles are enforced

$$x_{i-1,n_c} = x_{i,0},\ i = 2, ..., n_e \tag{3.17}$$

as well as the initial conditions $x_{1,0}{=}x_0$. Notice that the relation (3.17) holds since for Radau collocation points, $\tau_{n_c}{=}1$.

A significant difference between the Gauss-Lobatto and Radau collocation techniques is the order of the approach. Radau collocation requires that the last collocation point be placed at the end of the finite element. This reduces the order of the collocation method by one since it removes one degree of freedom from the method. Radau collocation has order $2n{-}1$ where n is the number of collocation points. Gauss-Lobatto collocation requires that the first and last collocation points be placed at the beginning and end of the finite element respectively. This gives Gauss-Lobatto collocation order $2n{-}2$. While there are benefits and disadvantages of both methods that will not be detailed here, it is important to emphasize that both methods proved to be reasonable discretization approaches for our applications.

3.4 Summary

Optimization of process systems has proven to be an effective method for improving operation and profits in the many industries (Scheu and Marquardt, 2011; Diehl et al., 2002; Zavala et al., 2008; Zhu et al., 2010; Tanaka and Martins, 2011). The success of these methods has led to the continued growth of these systems to improve model rigor and increase the scope of the optimization problem, however the solution of very large-scale models remains challenging (Hartwich and Marquardt, 2010). Furthermore, the successful use of advanced solution approaches requires that these algorithms be interfaced with effective problem formulation tools. Modern object-oriented modeling languages allow for rapid creation of complex optimization problems and reduce the burden of model development, optimization problem formulation, and solver interfacing, and while this eases the construction of complicated optimization problems, it also makes it easier to construct intractably large problems. To continue seeing dramatic improvements in solution times and feasible problem sizes, there is a need for the development of advanced parallel algorithms for dynamic optimization that can utilize parallel computing architectures and interface with modern modeling languages.

In the research presented here we demonstrate the application of interior-point methods to solve infectious disease parameter estimation problems that have been formulated as nonlinear programs. Because these problems can become very large and require the use of parallel solution approaches for efficient solution, we have also developed two parallel solution strategies for these problems and present those approaches in Sections 6 and 7.

4. ESTIMATING SEASONALITY WITH A DISCRETE-TIME INFECTIOUS DISEASE MODEL [*]

Infectious diseases remain a significant health concern around the world. Mathematical modeling of these diseases can be useful to better understand their dynamics and also to develop more effective control strategies. In this section, we show the capabilities of interior-point methods and nonlinear programming formulations to efficiently estimate parameters in multiple discrete-time disease models using measles case count data from three cities. These models include multiplicative measurement noise and have seasonality incorporated into multiple model parameters. Our results show that nearly identical seasonality can be estimated from different model parameters, and that the seasonality estimated in these parameters shows strong correlation to school term holidays even across very different social settings and holiday schedules.

The time-series SIR model from (Finkenstädt and Grenfell, 2000) is used as a basis for the model in our case studies because of its demonstrated ability to fit measles time-series data and produce estimates of seasonal transmission. Finkenstädt and Grenfell proposed a two-stage estimation approach which first estimated the susceptible dynamics and reporting rate using a susceptible reconstruction procedure (Finkenstädt and Grenfell, 2000). These estimates were then treated as known inputs to the second stage, where the parameters in the incidence function were estimated using a linearized model of transmission.

Instead of this two-stage approach, our nonlinear programming formulation es-

[*]Part of this section is reprinted with permission from "Interior-Point Methods for Estimating Seasonal Parameters in Discrete-Time Infectious Disease Models" by Word, D.P., Young, J.K., Cummings, D.A.T., Iamsirithaworn, S., and Laird, C.D., 2013. Submitted to PLOS One.

timates all the unknown quantities simultaneously (with constraints on the form of the reporting fraction). We consider the fully nonlinear model, not a linearized approximation, and the approach is flexible enough to address different model structures in a straightforward manner. In addition, it is important to note that this is not a one-step-ahead estimation, but instead, the entire time horizon is considered simultaneously.

In this section, the base problem formulation is introduced, along with a description of the sparse interior-point method used to solve the large-scale nonlinear problem. We describe three estimation formulations that incorporate seasonality into different model parameters. The first formulation estimates a seasonally varying transmission parameter β (labeled SVTP). The second formulation estimates a seasonally varying exponential parameter α (labeled SVEP). The third formulation estimates seasonality in the introduction of susceptibles into the population (labeled SVIS).

4.1 Estimation Problem Formulations

The deterministic skeleton of the TSIR model used in this analysis is given by,

$$I_{t+1} = \frac{\beta_{\tau(t)} I_t^{\alpha} S_t}{N_t^{\star}} \qquad \forall \ t \in T, \tag{4.1}$$

$$S_{t+1} = S_t + B_t^{\star} - I_{t+1} \qquad \forall \ t \in T, \tag{4.2}$$

where $t \in T$ refers to the set of discrete time periods, I_t is the number of new cases in time period t, S_t is the current number of susceptible individuals, and $B_t^{\star}$ corresponds to the number of yearly births divided by the number of discrete-time intervals per year. Equation 4.1 describes the infection process and Equation 4.2 is the susceptible balance. Note that the discrete time interval is assumed to be the same as the generation time of disease and, as such, removes the distinction between incidence and prevalence. Full recovery is assumed from one time interval to the next. The transmission parameter $\beta_{\tau(t)}$ is seasonal and restricted to be the same from year to year (i.e. $\tau(t)$ is a mapping from the overall time interval t to an index from the beginning of the current year only). The exponent α is a parameter that allows a nonlinear dependence on I_t in the incidence function (Finkenstädt and Grenfell, 2000; Liu et al., 1987; Finkenstädt et al., 2002; Bjornstad et al., 2002; Grenfell et al., 2002). In this model, population $N_t^\star$ and births $B_t^\star$ are known inputs.

The goal is to estimate the unknown parameters $\beta_{\tau(t)}$ and α along with the unobserved state S_t using reported incidence. However, the cases are almost always under-reported. Therefore, the true incidence I_t is related to the reported incidence $C_t^\star$ by an unknown, potentially time-varying, reporting fraction γ_t,

$$C_t^\star = \gamma_t I_t. \tag{4.3}$$

In the absence of additional information or further restriction of the time-varying reporting fraction, it is clear that unique estimation is not possible. Any value for I_t can be matched exactly to the reported incidence $C_t^\star$ by setting $\gamma_t{=}I_t/C_t^\star$. In our work, we will assume that the reporting fraction varies linearly over the entire time horizon, although this framework supports general restrictions on its functional form. Additionally, we assume multiplicative measurement noise in the reported cases since the variance of the noise appears to increase with the number of reported cases (Finkenstädt and Grenfell, 2000) (i.e., $C_t^\star = \gamma_t I_t \epsilon_t^C$ where ϵ_t^C is an unknown error term).

To improve the scaling and convergence properties of the nonlinear estimation formulation, an exact log transformation is performed on the incidence expression and the reporting fraction correction expression. This gives the formulation of our discrete-time deterministic model with seasonally varying transmission parameter (SVTP), β, shown in Problem 4.4.

$$\min \quad \sum_t (\tilde{\epsilon}_t^C)^2 \tag{4.4a}$$

$$\text{s.t.}$$

$$S_{t+1} = S_t + B_t^\star - I_{t+1} \qquad \forall \; t \in T_- \tag{4.4b}$$

$$\tilde{I}_{t+1} = \tilde{\beta}_{\tau(t)} + \alpha \tilde{I}_t + \tilde{S}_t - \tilde{N}_t^\star \qquad \forall \; t \in T_- \tag{4.4c}$$

$$\gamma_{t+1} = \gamma_t + \gamma_{inc} \qquad \forall \; t \in T_- \tag{4.4d}$$

$$\tilde{C}_t^\star = \tilde{\gamma}_t + \tilde{I}_t + \tilde{\epsilon}_t^C \qquad \forall \; t \in T \tag{4.4e}$$

$$I_t = \exp(\tilde{I}_t) \qquad \forall \; t \in T \tag{4.4f}$$

$$S_t = \exp(\tilde{S}_t) \qquad \forall \; t \in T \tag{4.4g}$$

$$\gamma_t = \exp(\tilde{\gamma}_t) \qquad \forall \; t \in T \tag{4.4h}$$

$$\beta_y = \exp(\tilde{\beta}_y) \qquad \forall \; y \in T_y \tag{4.4i}$$

$$0 \leq \gamma_t \leq 1 \qquad \forall \; t \in T \tag{4.4j}$$

$$0 \leq I_t, S_t \leq N_t^\star \qquad \forall \; t \in T \tag{4.4k}$$

Here, $t \in T$ refers to the discrete time interval in the entire time horizon T, set T_- is identical to set T except that it is missing the last element of T, and T_y is the set of time intervals within a single year. I_t is the number of new cases at time t, $C_t^\star$ is the number of reported cases at time t, S_t is the number of susceptible individuals at time t, $B_t^\star$ still corresponds to the number of yearly births divided by the number of discrete-time intervals per year, and $N_t^\star$ is the population at time t. The variable $\tilde{\epsilon}_t^C$ is the log transformation of the multiplicative error in the number of reported cases,

α is an exponential parameter on the incidence, and γ is the incidence reporting fraction that is assumed to vary linearly by some increment γ_{inc} between each time interval. The seasonal transmission parameter $\beta_{\tau(t)}$ is restricted to be the same from year to year (i.e. $\tau(t)$ is a mapping from the overall time interval t to an index y from the beginning of a single year only). The $\sim$ symbol denotes log-transforms such that $\tilde{I}_t$, $\tilde{S}_t$, $\tilde{\gamma}_t$, $\tilde{\beta}$, and $\tilde{C}_t^\star$ are the log-transformations of I_t, S_t, γ_t, β, and $C_t^\star$ respectively. Note that this model uses exact log transformations and not linear approximations.

This formulation has several advantages over previously existing approaches for discrete-time models. This approach simultaneously estimates the susceptible dynamics and the reporting fraction along with the disease parameters, and provides an estimate of the susceptible count profiles in time. In addition, this formulation can easily account for missing data by removing terms from the objective function for periods where no data is available, and given the flexible nature of the framework, a variety of measures of fit could be used as the objective function.

The estimation formulation (SVTP) above assumed seasonality in the transmission parameter β, however, it is reasonable to postulate models with seasonality in other model parameters. In an effort to better understand potential drivers of observed infectious disease dynamics, this paper also presents estimation results for formulations that include unknown seasonality in the exponential parameter α, and in the birth rate. In particular, we wish to know if alternate models provide improved fit to the data, and if the estimated seasonal patterns are the same for different parameters (e.g., are they still correlated with school holiday schedules). The first alternative model formulation includes a time-invariant transmission parameter but seasonal exponential parameter α. This formulation is identified as SVEP and is identical to that shown in Equation 4.4 except that the incidence balance (4.4c) is now

$$\tilde{I}_{t+1} = \tilde{\beta} + \alpha_{\tau(t)}\tilde{I}_t + \tilde{S}_t - \tilde{N}_t \qquad \forall\ t \in T_-. \tag{4.5}$$

Here, β is no longer seasonal, but α is defined to be seasonal where $\tau(t)$ is a mapping from the overall time interval t to an index from the beginning of the current year only.

In the third formulation, we investigate estimation of seasonality in the birth rate. Our available birth data includes the yearly number of births only, and in previous formulations, we have assumed that births occur uniformly throughout the year. However, births may contribute to susceptible dynamics in a non-uniform way throughout the year due to the newborn children effectively entering the pool of individuals at risk of infection at particular times of year (e.g. school entry). Therefore, we developed a formulation that estimates unknown seasonality in the birth data from case count data to see if seasonality observed in infection data can be captured by seasonally varying births.

The model formulation assuming seasonality in births, identified as SVIS, incorporates several differences from the previous formulations. Rather than assuming a uniform addition of births into the susceptible population throughout the year, this formulation includes a weighting factor ν_τ that allows for seasonal variation in births, keeping β and α time-invariant. This model formulation differs from that in Equation 4.4 by replacing the susceptible balance (4.4b) with (4.6), the incidence balance (4.4c) with (4.7), and adding one new constraint (4.8),

$$S_{t+1} = S_t + \nu_{\tau(t)}B_t^\star - I_{t+1} \qquad \forall\ t \in T_- \tag{4.6}$$

$$\tilde{I}_{t+1} = \tilde{\beta} + \alpha\tilde{I}_t + \tilde{S}_t - \tilde{N}_t \qquad \forall\ t \in T_- \tag{4.7}$$

$$\sum_{i \in T_y} \nu_{\tau(t)} = |T_y|. \tag{4.8}$$

Here, T_y refers to the set of discrete time within a single year, $\nu_{\tau(t)}$ is a seasonally varying weight on births, and $B_t^\star$ correspond to the number of yearly births divided by the number of discrete-time intervals per year. Equation 4.8 ensures that the number of new susceptibles introduced into the population every year is equal to the number of reported births for each year, and $|T_y|$ is the cardinality of T_y (i.e., the number of discrete time intervals per year). However, this constraint allows these new susceptibles to be added in a seasonally varying manner.

In this section, we estimate parameters using the large-scale, full-space interior-point method, Ipopt (Wächter and Biegler, 2006; Laird and Biegler, 2008). The Ipopt algorithm implements a primal-dual log-barrier interior-point approach for handling large-scale nonlinear (and non-convex) programming problems that may have many variable bounds. The Ipopt algorithm makes use of full first and second order derivative information for the constraints and the objective. In this research, the modeling language AMPL (Fourer et al., 1993) was used to describe the problem formulation. AMPL provides efficient numerical values of the analytical derivatives through automatic differentiation. The original implementation of the algorithm was developed in Fortran by Andreas Wächter and Lorenz T. Biegler, and for full details of the algorithm please see the literature (Wächter, 2002; Wächter and Biegler, 2006).

4.2 Description of Data Used in Estimation

Four different data sets were considered in this work. Simulated data from an SIR model was used to validate the estimation procedure. Case data from London was used to compare our estimation results with existing literature values. The London data set has been heavily studied and is used in this work to demonstrate the agreement of our approach with existing literature. Data from two cities, New York City (NYC) and Bangkok, are investigated in this study using all 3 estimation formulations. The NYC and Bangkok data sets are used due to their very different school term holidays, allowing us to show the correlation between school terms and seasonal transmission. NYC has a long summer school holiday lasting from the end of June until mid September, while Bangkok has two long school holidays: one from the beginning of March until the middle of May and one the entire month of October.

The data from London reports biweekly measles cases and yearly birth rate data for the years 1944-1963 (Bjornstad et al., 2002). A constant population was assumed for this data set. The data from New York City (NYC) contained yearly reported population and birth rate data, and monthly reported measles case counts for the years 1944-1963 (Yorke and London, 1973). The Bangkok data contained yearly reported birth rate data (of Epidemiology, 1986), but the population was only reported every decade. Linear interpolation was used to approximate the yearly populations across the time horizon studied. The measles case counts were reported monthly for the years 1975-1986. In all of these estimations, the population is assumed to vary linearly throughout each year, and the birth rate is assumed to be uniform throughout each year.

In our estimation approach, we assume that the under-reporting of incidence varies linearly in time, and we estimate a reporting fraction along with other model parameters. An additional challenge in the Bangkok data is the absence of case count data for the year 1979. To account for this, the formulation is modified to exclude these points from the objective function, while still including them in the simulated dynamics.

To estimate the discrete-time model, data must be available on the same time interval as the model discretization. The London data is available in a biweekly form that is consistent with the model discretization. However, the NYC and Bangkok data were reported monthly, so the data must be converted into a biweekly form. To resample the data, the monthly case counts are first converted to cumulative case counts. This cumulative data is interpolated at biweeks with a piecewise cubic Hermite interpolating polynomial (using the pchip method in MATLAB) to ensure no overshoot. The number of new cases in the biweekly intervals was then estimated by taking the difference between the biweekly interpolated data points.

4.3 Estimation Results

In this section, we present estimation results from the three different formulations. All models assumed multiplicative noise in the measurements but differed in how seasonality was included in the model. First, we validated our estimation procedure on simulated data using a model incorporating the common assumption of seasonality in the transmission parameter β. Additionally, we perform estimation on real measles case count data from London and compare our results with other literature studies. After validating our estimation procedure, we present estimates for seasonal profiles using measles data from New York City and Bangkok. These two data sets are used for estimates with all three model formulations.

Confidence intervals and regions were found using the log-likelihood method presented in Rooney and Biegler (Rooney and Biegler, 2004). Confidence intervals for all estimated parameters were constructed by fixing one parameter and allowing optimization over the remaining parameters. Confidence regions were created by fixing the 2 parameters being compared and optimizing all other variables. These confidence regions show the relationship between the exponential parameter (α), the mean of the transmission parameters ($\bar{\beta}$), and the mean susceptible fraction ($\bar{S}/N$). Appendix G contains the AMPL code used to generate these confidence regions.

4.3.1 Model and Procedure Validation

We first tested the SVTP (4.4) estimation formulation using known parameter values. We performed 10,000 simulations with an SIR model using MATLAB. Our simulations used a constant population of 10,002,000, a birth rate of 2.5% of the population per year, and a reporting fraction of 0.5. To generate 20 years of case data, the deterministic model was simulated for 100 years to achieve a cyclic steady state, and the final values from this simulation were used as the initial values for the 20 year simulation. The simulations were performed with the same model as that used for the estimation. Multiplicative measurement noise was drawn from a log-normal distribution with mean 1 and a standard deviation of 0.1 and applied to the reported cases.

Figures 4.1 and 4.2 demonstrate that our estimation approach gives a good estimate for β using data from the SIR simulations. In Figure 4.1, the circles show the true parameter values used for all 10,000 simulations. The solid line shows the mean of the estimated values for the parameters and the dashed lines show the 2.5 and 97.5 quantiles for the parameters estimated from all 10,000 simulations (giving 95% confidence intervals for these estimates). Figure 4.2 shows the estimates for a single simulated data set randomly selected from the pool of 10,000 simulations. The solid line shows the estimated parameters while the true parameters are shown with circles. The dashed lines show the 95% confidence intervals computed for this single estimation. The true parameter values are included inside these confidence intervals.

To further validate this approach, we then performed estimations with the London data that had been used in other studies and found our results to be consistent with other literature estimates (Fine and Clarkson, 1982; Grenfell et al., 2002; Finkenstädt et al., 2002; Cauchemez and Ferguson, 2008). Our estimate of seasonality in β for

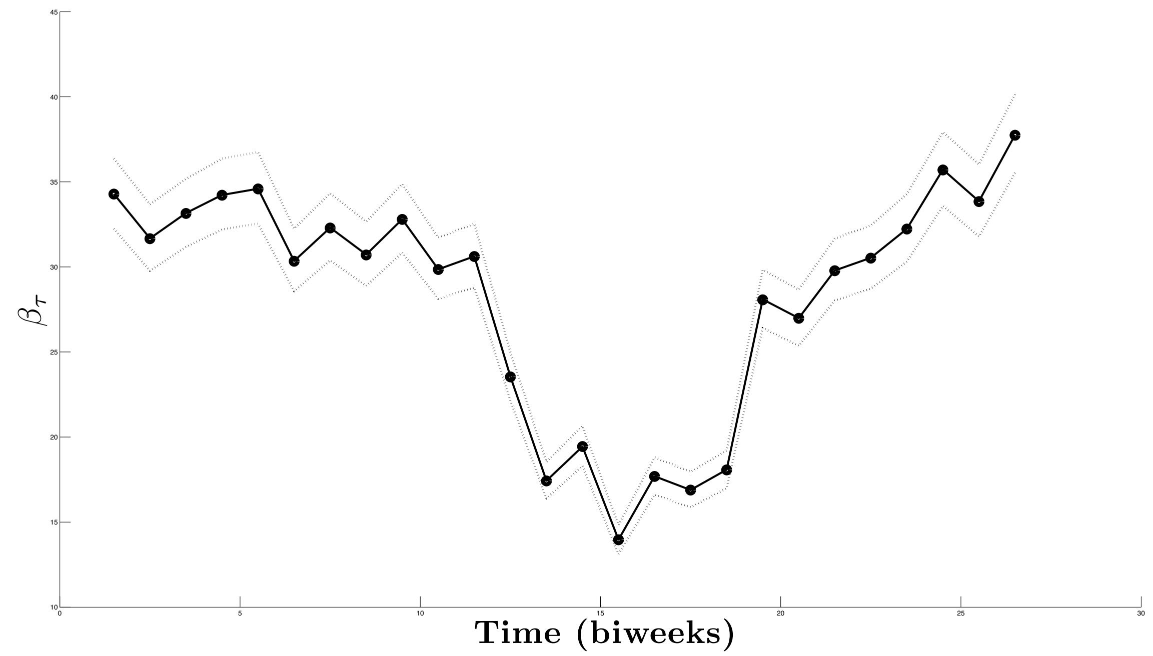

Figure 4.1: The true values of β used in the SIR simulation study (circles). The mean of the estimates from the simulation study (solid line). The 2.5th and 97.5th quantiles of the estimates from the simulation study (dashed lines).

London is similar to that obtained by Finkenstädt and Grenfell (Finkenstädt and Grenfell, 2000) for England and Wales over the same time period. Figure 5.6 shows a comparison of our estimated β with that estimated by Finkenstädt and Grenfell (Finkenstädt and Grenfell, 2000). The seasonality observed shows a small drop in the transmission parameter at the Easter break (biweek 8), and a large drop at the summer break (biweeks 15-18). This observation is in agreement with the proposal that transmission of measles is correlated with school holidays.

Our estimates are very similar to others in the literature, and our solution approach is also very fast. The run time for the London estimation was under 5 seconds even though this estimation included 21 years of data and estimated the susceptible population and reporting fraction simultaneously with the model parameters. The computational time required for all estimates reported in this work using real case

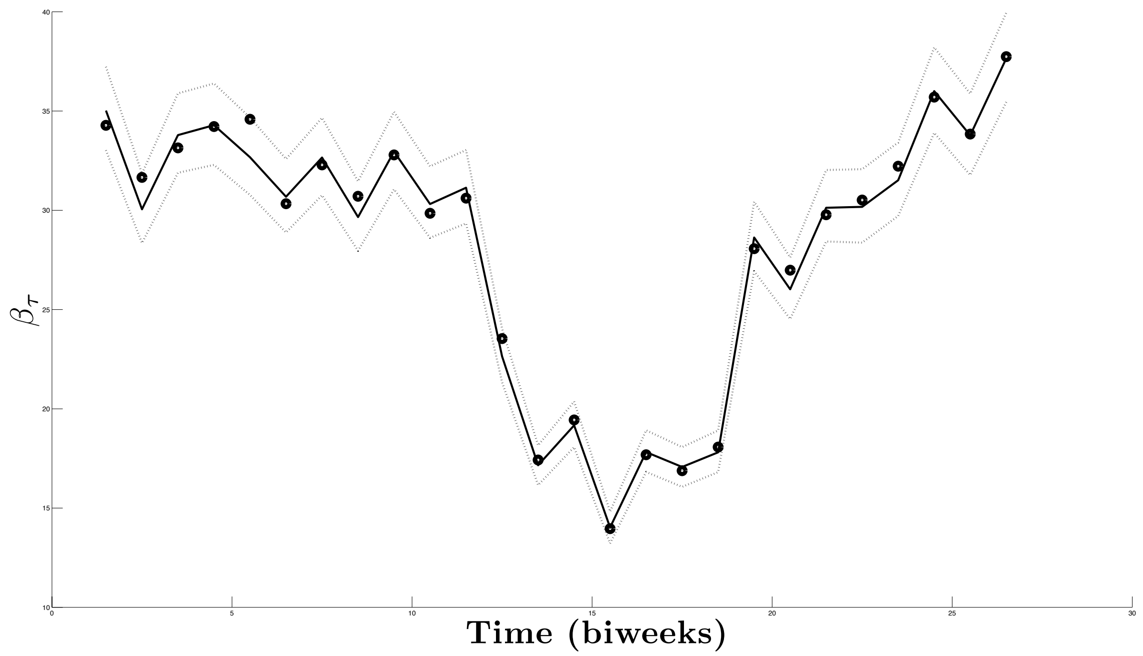

Figure 4.2: The estimated transmission profile β (solid line) for a single data set with 95% confidence intervals (– –) found using log-likelihoods as described in Rooney and Biegler (2004). The true values of β used in the SIR simulation (circles).

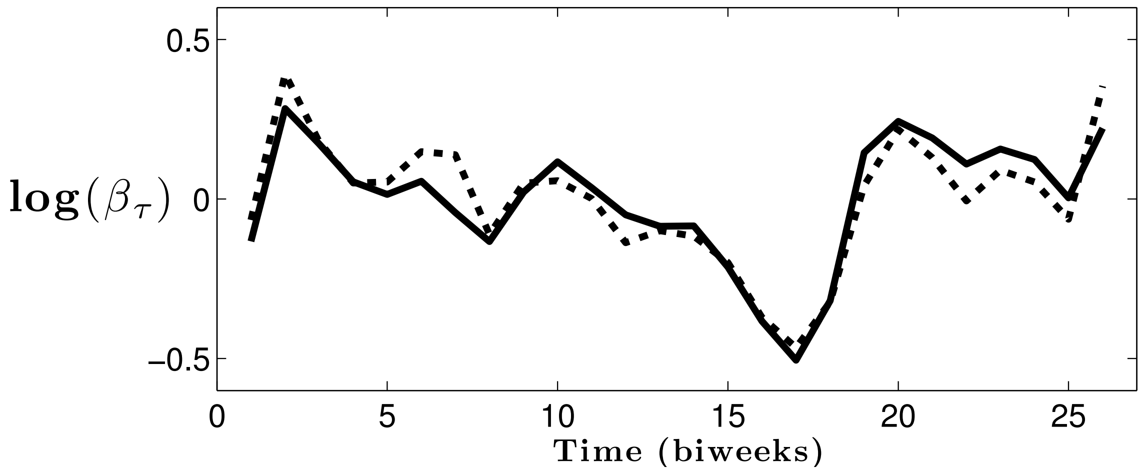

Figure 4.3: A comparison of seasonal transmission parameters estimated for London. The values estimated in this work (– –) are overlaid with those reported by Finkenstädt and Grenfell (2000) (—).

data is given in Table 4.1. Given our ability to accurately estimate known parameters given simulated data, and the similarity between our estimates and other estimates from the literature using London data, we are confident that our estimation procedure offers a fast, reliable method to estimate seasonal parameters in infectious disease models.

City	Model	Variables	Constraints	CPU Time (sec)[a]
London	β_τ	3878	3847	4.5
NYC	β_τ	3696	3665	2.3
	α_τ	3671	3640	3.2
	ν_τ	3671	3640	3.2
Bangkok	β_τ	2240	2183	0.6
	α_τ	2215	2158	5.6
	ν_τ	2215	2158	0.6

Table 4.1: Problem size and solution times for the London, New York City (NYC), and Bangkok estimation problems studied in this paper.

4.3.2 Estimates of Seasonality in Different Model Parameters

In this section, we show the estimation results for the three different problem formulations SVTP, SVEP, and SVIS, addressing seasonality in the transmission parameter, the exponential parameter, and the introduction of susceptibles respectively. Each of these three formulations is solved using measles data from both New York City and Bangkok, two locations with significantly different school holiday schedules.

The first set of estimation results are shown for formulation SVTP using measles data from New York City. Our estimates using this data yield an estimated number of reported measles incidence that fits reasonably with the actual reported incidence data as shown in Figure 4.4. Recall that this is not a one-step-ahead estimation. The mismatch shown at the beginning of the time horizon may be due to our assumption that the reporting fraction varies linearly in time. The biennial periodicity seen in the case counts and the estimated number of susceptibles (Figure 4.5) is consistent with expectations for endemic measles in cities with a low birth rate. Our estimate of β using NYC measles data is shown in Figure 4.6, and the observed seasonality coincides strongly with the school term summer break which occurred over biweeks 11-17. While the confidence intervals shown in Figure 4.6 seem large, recall that we determine these intervals by fixing the value of a single parameter and optimizing over the remaining parameters (i.e., re-estimating the remaining parameters) to generate profile likelihoods.

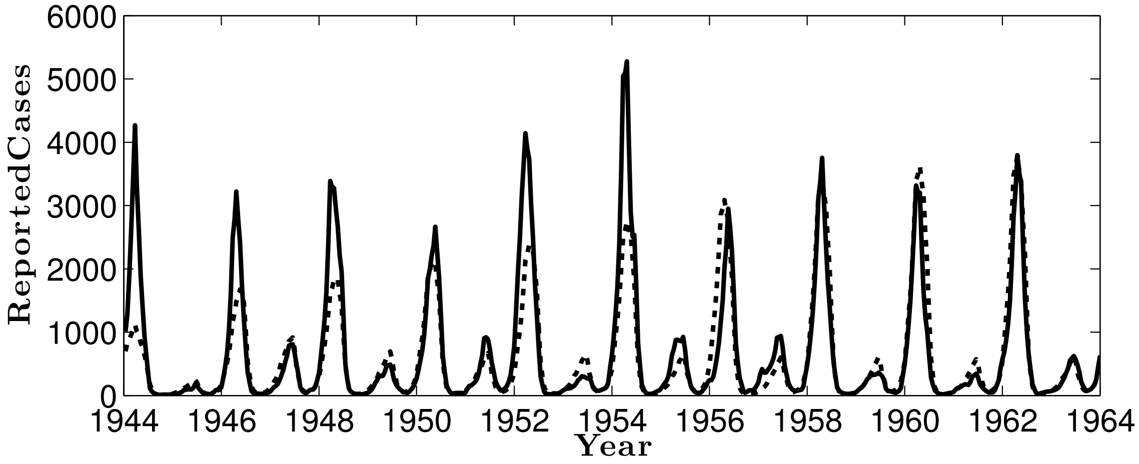

Figure 4.4: New York City results: The estimated number of reported cases (– –) with the actual number of reported cases (—) of measles.

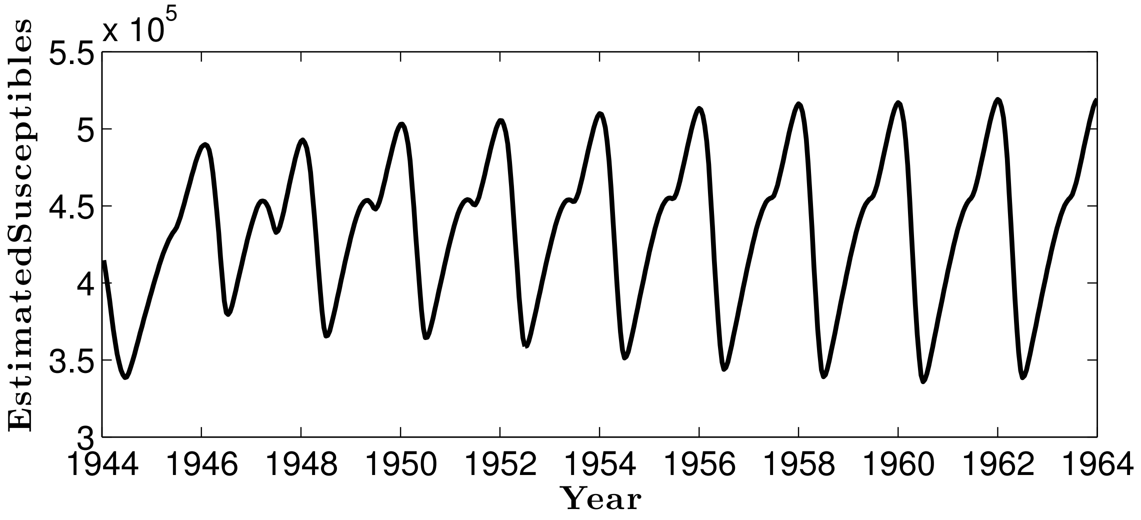

Figure 4.5: New York City results: The estimated number of individuals susceptible to measles.

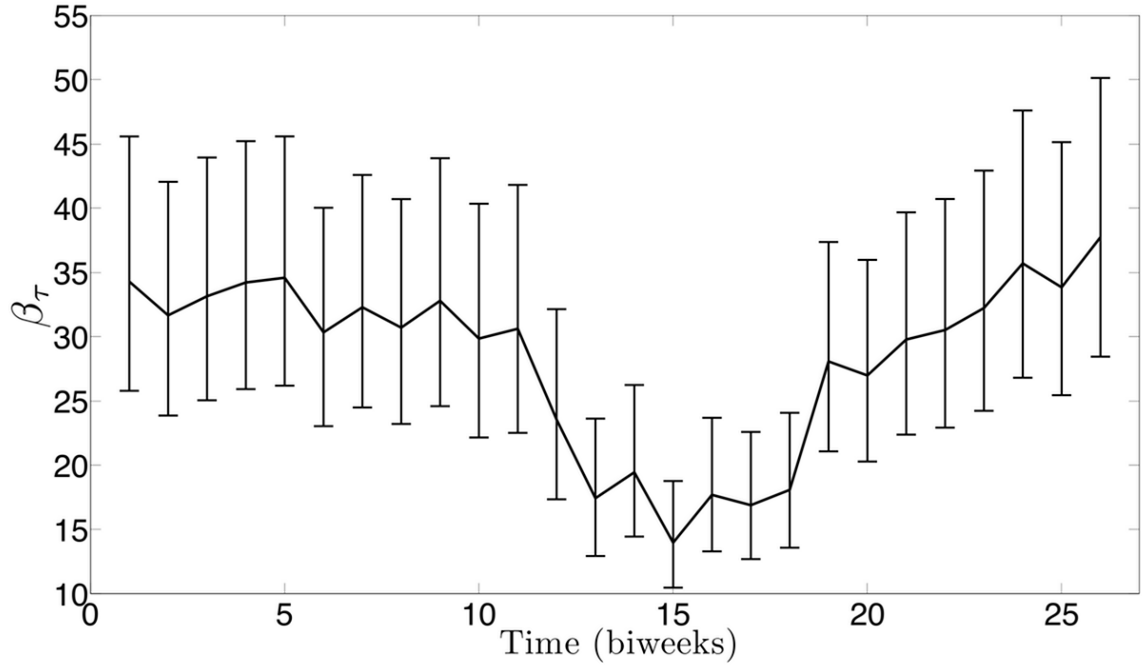

Figure 4.6: Estimated β_τ for measles in New York City with 95% confidence intervals (—).

At the solution of the estimation problem, the optimization package Ipopt indicates that the reduced-Hessian is positive definite, which implies that the estimated parameters are locally unique. However, in addition to single variable confidence intervals, we are also interested in confidence regions where parameter values can be expected. Using the procedure outlined in Rooney and Beigler (Rooney and Biegler, 2004), we construct pairwise confidence regions for the mean of the transmission parameter, the mean of the susceptible population, and the exponential parameter α, based on the likelihood ratio test. Figure 4.7 shows the relationship between the mean of the transmission parameters ($\bar{\beta}$) and the mean susceptible fraction ($\bar{S}/N$), Figure 4.8 shows the relationship between the mean of the transmission parameters ($\bar{\beta}$) and the exponential parameter (α), and Figure 4.9 shows the relationship between the exponential parameter (α) and the mean susceptible fraction ($\bar{S}/N$). In all figures the plus sign indicates the optimal solution, and the bold line indicates the extent of the 95% confidence region. All three regions have similar shapes and show some correlation between the parameters. High values of $\bar{\beta}$ correspond to lower values of $\bar{S}$ and α. This is reasonable since an increase in the infection term $\beta I^{\alpha} S$ caused by a higher β could be offset somewhat by a reduction in either α or $\bar{S}$. An increase in α corresponds to a higher value of $\bar{S}$ which seems contrary to the previous results, however, when constructing these intervals, we optimize over the remaining variables and a lower value of α corresponds to a higher value of $\bar{\beta}$. Furthermore, the relative range of α in the confidence region is much smaller than that of $\bar{\beta}$ and $\bar{S}$. While not shown here for brevity, confidence regions with similar characteristics were found for the other data sets used in this section.

The Bangkok data allows us to perform estimates for a location with a very different social environment and school schedule than NYC. These case counts suffered from a much lower reporting fraction and even missing data during one year (1979).

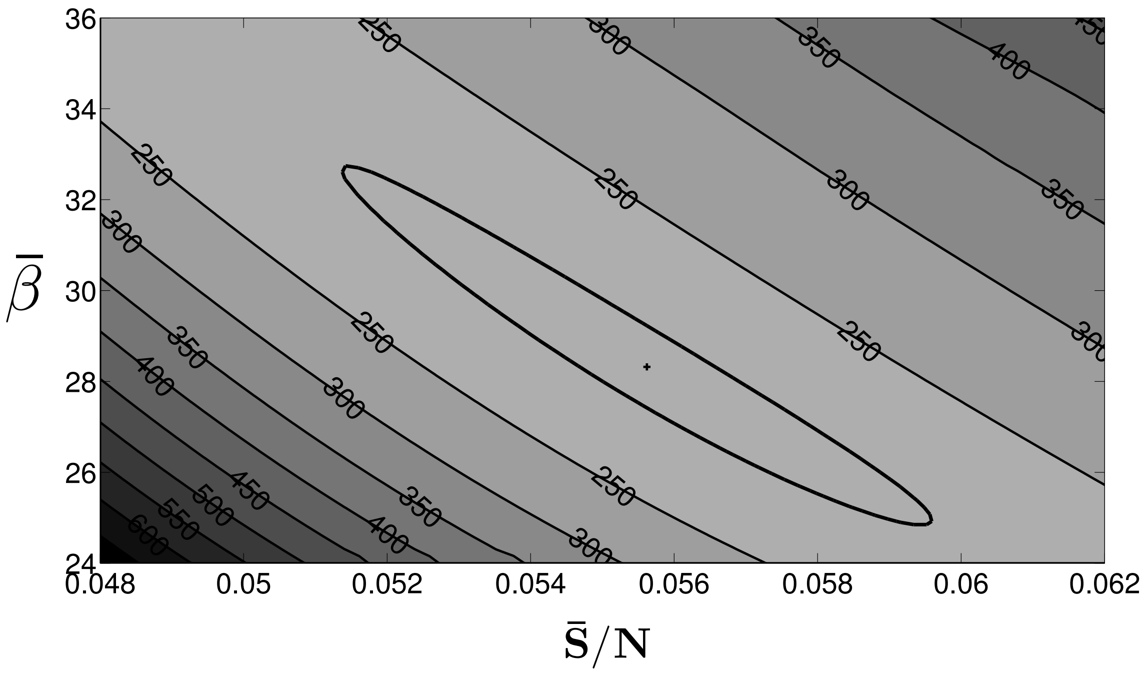

Figure 4.7: New York City results: 95% confidence region for $\bar{\beta}$ and $\bar{S}/N$.

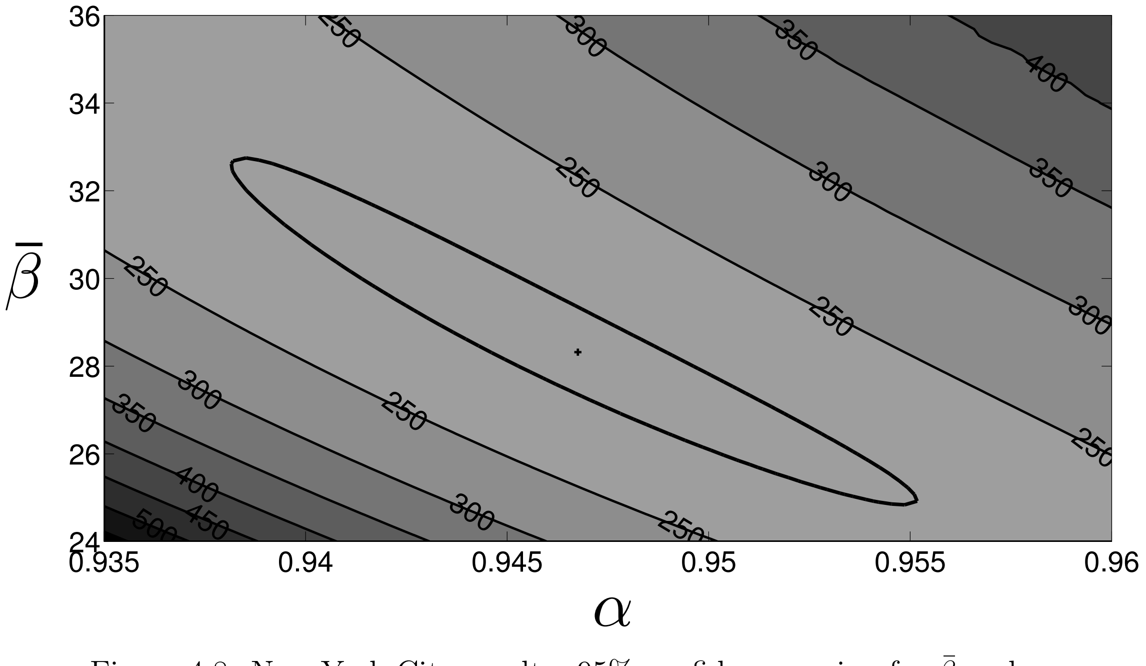

Figure 4.8: New York City results: 95% confidence region for $\bar{\beta}$ and α.

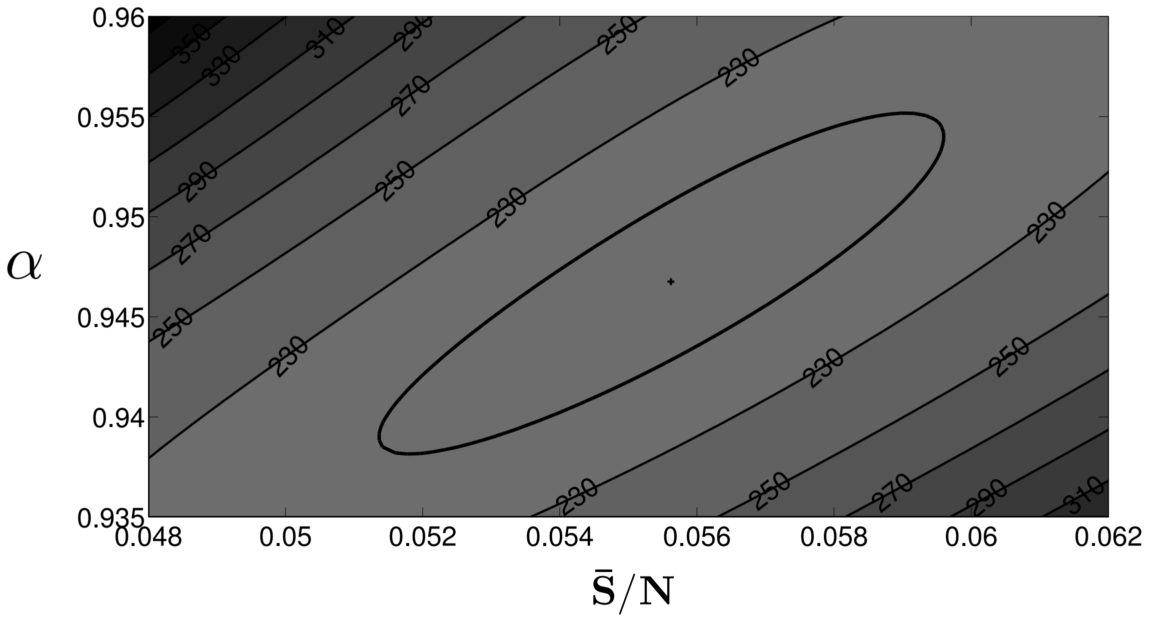

Figure 4.9: New York City results: 95% confidence region for α and $\bar{S}/N$.

Still, the estimated reported measles incidence gives a remarkably good fit to the actual reported incidence data as shown in Figure 4.10.

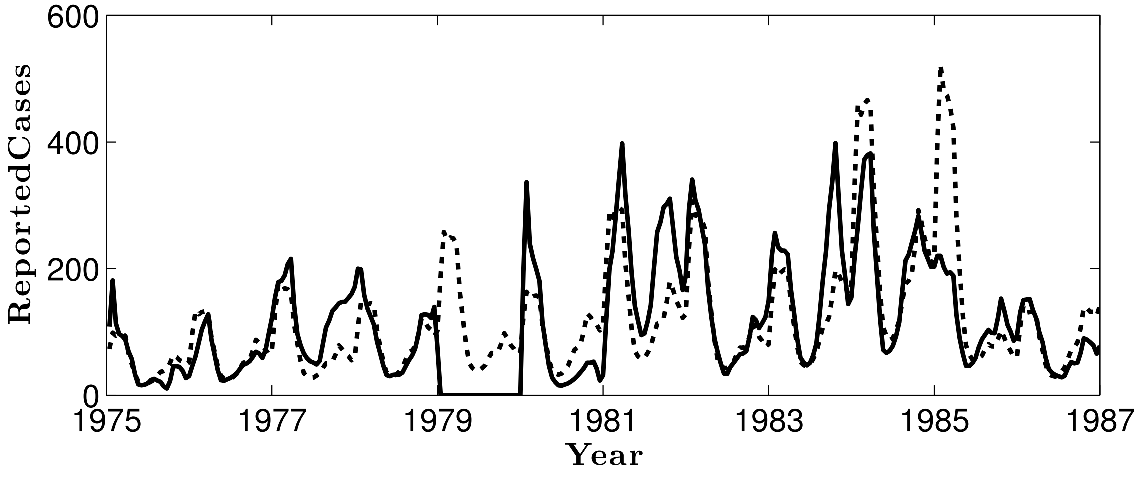

Figure 4.10: Bangkok results: The estimated number of reported cases (– –) with the actual number of reported cases (—) of measles. Note: Case data is unavailable for 1979.

Our estimate of β for Bangkok is shown in Figure 4.11, and, as expected, the estimated transmission profile is very different from those estimated for both London and NYC. However, the observed seasonality again appears to be correlated with the school term holidays that occur from the beginning of March through the middle of May and the entirety of October (corresponding approximately to biweeks 5-9 and 20-21 respectively). This estimated seasonality does not appear to be as strong as that seen in the NYC estimate, but this could be due to the high degree of under-reporting in this data set. Our estimates show that only about 1% of cases are reported at the beginning of the time horizon and that this fraction increases to only about 5% of the cases being reported by the end of the time horizon. This low reporting fraction allows for significant noise to be present in the available data which could reduce our ability to estimate seasonality. The complete set of parameter estimation results and confidence intervals for the SVTP estimates using NYC and Bangkok data are given in Table 4.2.

In an effort to investigate seasonality in other model parameters, the SVEP model formulation was used to estimate seasonal exponential parameters α using data from both New York City and Bangkok. This formulation includes a time-invariant transmission parameter and seasonal exponential parameters α (4.5). Similar to the estimated profiles using SVTP, these estimations show a seasonal profile for α that appears strongly correlated with the school holiday schedule.

The seasonal α estimated using the New York City measles data is shown in Figure 4.12, and the seasonal α estimated using the Bangkok measles data is shown in Figure 4.13. After scaling, the estimated seasonality in α is almost identical to the profiles estimated for the seasonal β's. This demonstrates that while seasonality provides a mechanism to capture the dynamics seen in the data, the actual implementation of the seasonality into the model can vary. This also shows that α and β could be

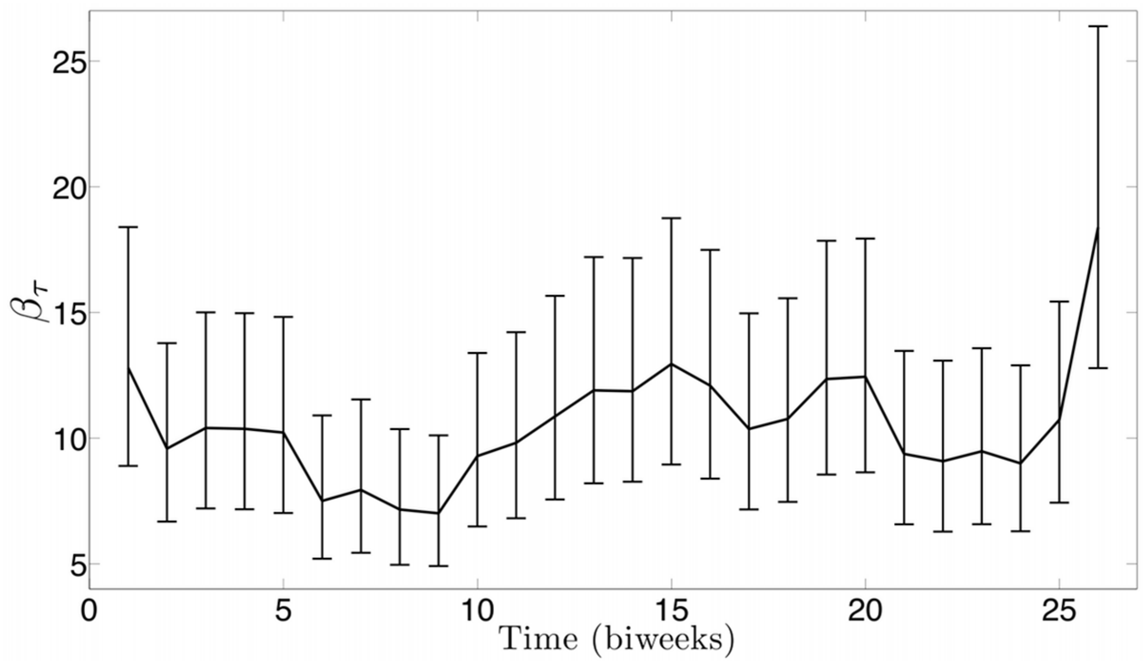

Figure 4.11: Estimated β_τ for measles in Bangkok with 95% confidence intervals (–).

describing some combination of several physical phenomena, and care must be taken when interpreting the underlying cause of the seasonality.

To compare these results numerically, Spearman correlation coefficients were computed to compare our estimated seasonality with school holidays. For the school term profile we construct a 0/1 sequence where a 1 indicates that school is in session for a particular biweek and a 0 indicates that school is on holiday. Figure 4.14 shows the Spearman correlation coefficient computed using reported holiday schedules and the estimated seasonal β parameters for New York City. We also computed the Spearman correlation coefficient using a holiday schedule that is shifted forward by one biweek, and this coefficient is even higher. This shift is not unreasonable given that the data was reported monthly but was resampled into a biweekly form suitable for our formulation. The histogram in this figure shows the distribution of correlations that were computed between the reported holiday schedule and 1,000 randomly or-

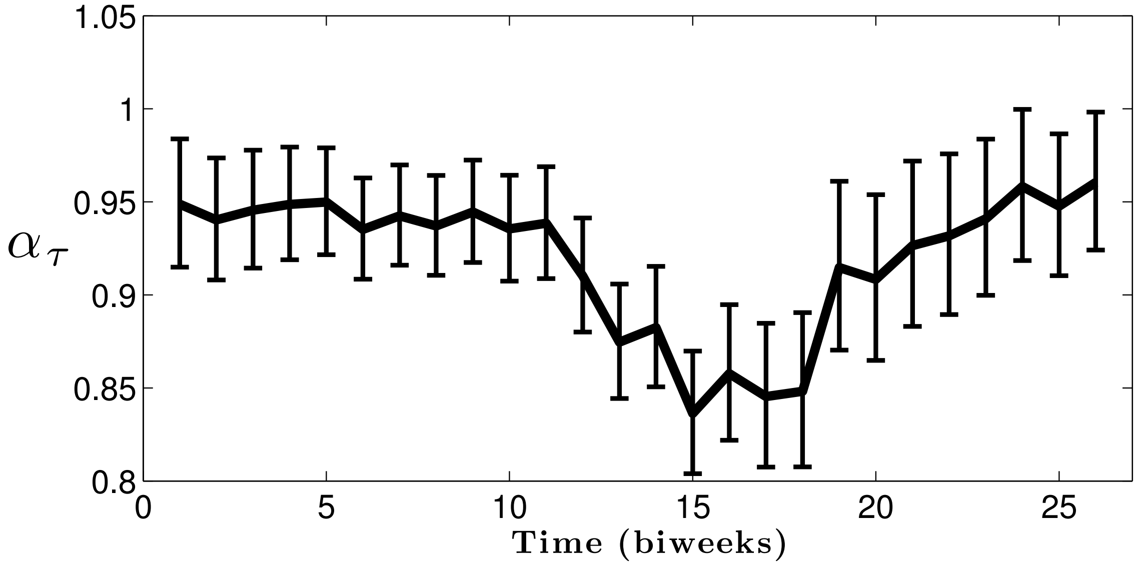

Figure 4.12: New York City estimates of seasonal mixing parameter α_τ with 95% confidence intervals.

dered vectors from our estimated β values. This histogram demonstrates that the correlation between school holidays and random seasonality in the parameters is normally distributed about zero, while the correlation between the holidays and our estimated seasonality is very strong (above the 95% confidence level). Figure 4.15 displays the same calculations as Figure 4.14 except using Bangkok school holidays and parameter estimates.

The same calculations were performed using the estimated seasonal α values with almost identical results. The Spearman correlation coefficients for New York City and Bangkok using the unshifted holidays and seasonal α's was 0.62 and 0.61 respectively. Using shifted holidays, these correlation coefficients were 0.76 and 0.72 for New York City and Bangkok respectively. These results strongly support our belief that our estimated seasonality is correlated with school holidays. Furthermore, the estimated seasonality in α and β are strongly correlated with each other. The correlation coefficient between the estimated seasonal α and seasonal β for New York City was

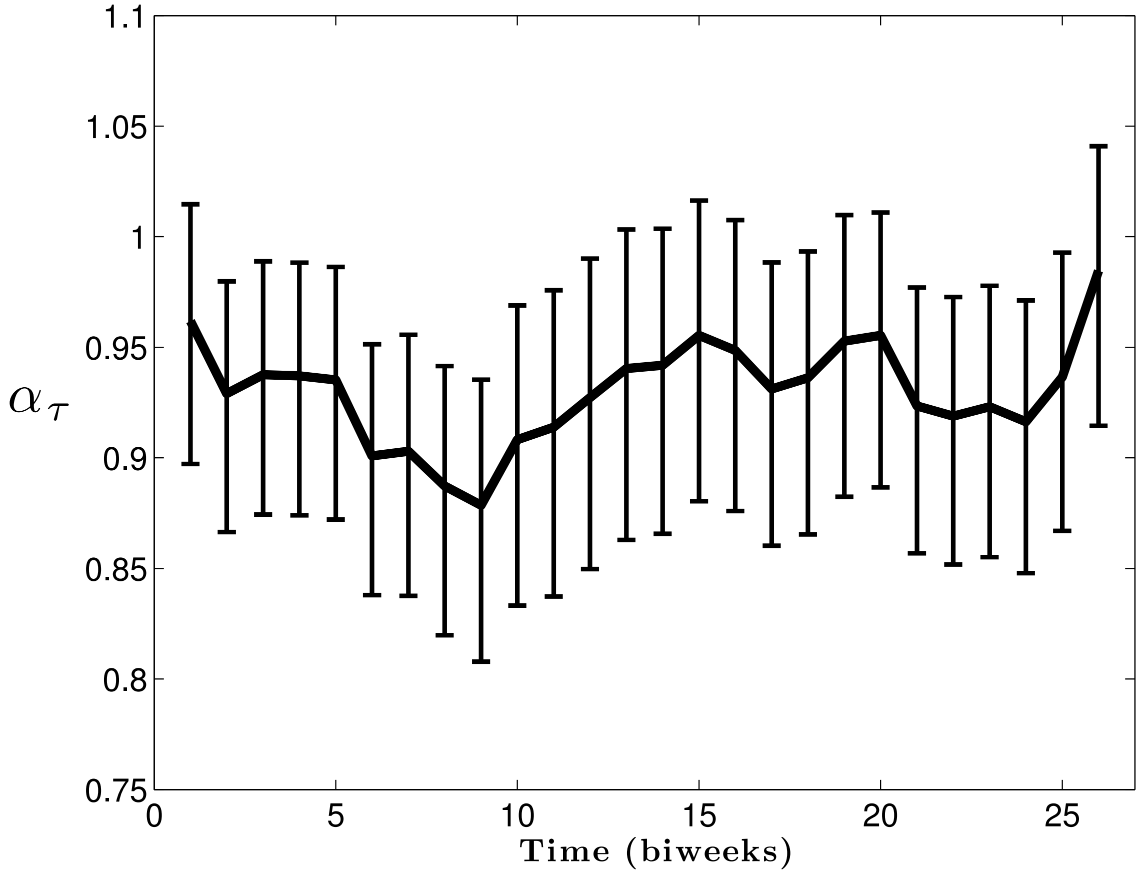

Figure 4.13: Bangkok estimates of seasonal mixing parameter α_τ with 95% confidence intervals.

0.99, and the correlation coefficient between the estimated seasonal α and seasonal β for Bangkok was 0.97.

For the time periods under consideration, we have only yearly birth data. For the previous estimation results (SVTP and SVEP) we assumed that the birth rates were constant throughout the year. Formulation SVIS estimates seasonality in the introduction of susceptibles into the S compartment. The SVIS model formulation was used for estimates using data from New York City and Bangkok. This formulation contains time-invariant transmission and exponential parameters and considers seasonality in births by including a weighting factor ν_τ that must be estimated.

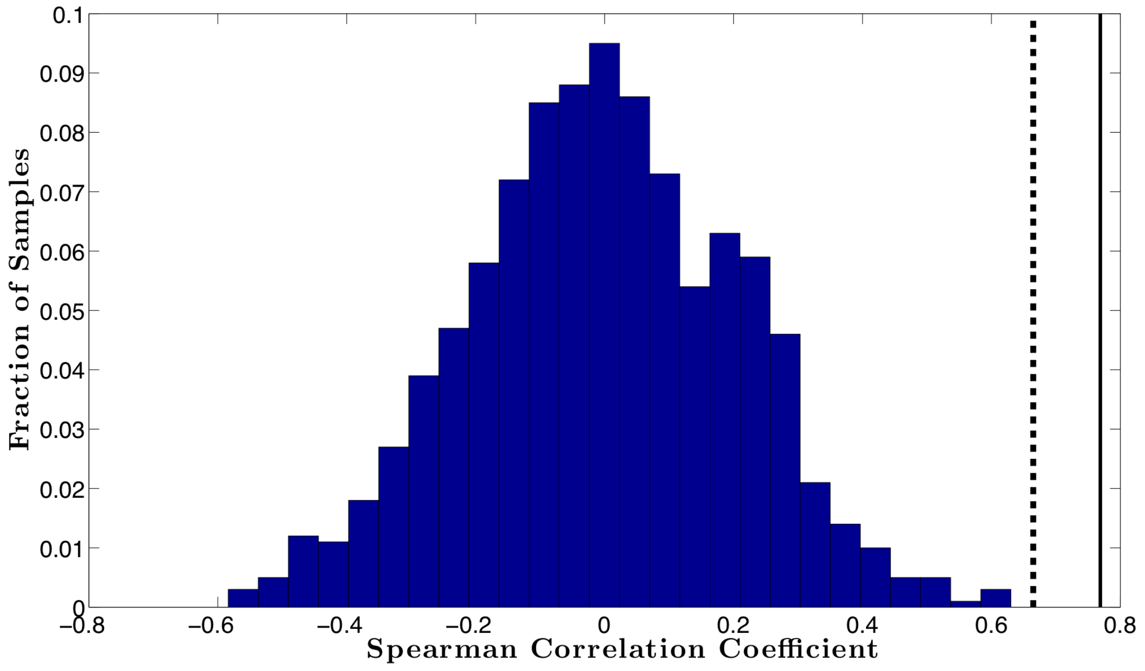

Figure 4.14: Spearman correlation coefficients computed for New York City are shown. The coefficient calculated using reported holiday schedules and estimated parameters are shown by the dashed line. The coefficient computed using a holiday schedule shifted forward by one biweek is shown by the solid line. The histogram shows the distribution of correlations that were computed between the reported holiday schedule and 1,000 randomly ordered vectors of our parameter estimates.

The seasonal profile for ν estimated using the New York City measles data is shown in Figure 4.16, and the seasonal profile for ν estimated using the Bangkok measles data is shown in Figure 4.17. Just as with previous estimates, these results show strong correlation between the seasonality observed in measles case data and school term holidays. For New York City, ν is essentially zero except at the end of September which is immediately following the start of the fall semester of school. For Bangkok, ν is essentially zero except at the beginning of June. These results do not show a seasonal variation that is consistent with estimated seasonal profiles for the other two parameters, however, these profiles are very interesting in that they are still highly correlated with the school schedules in both settings. The seasonal

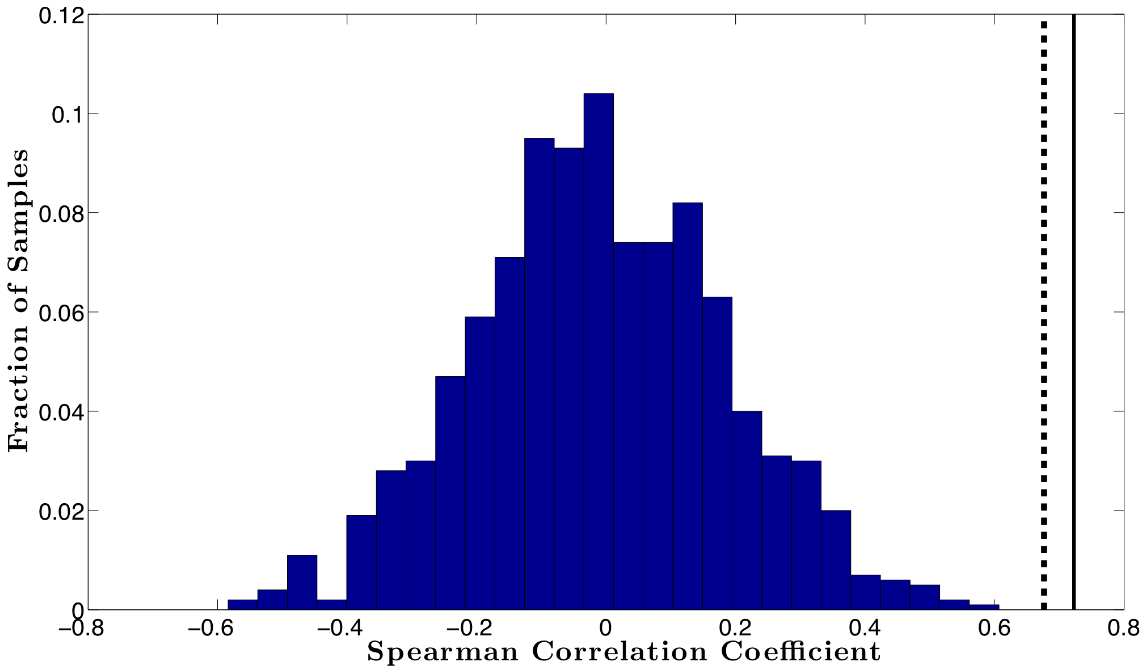

Figure 4.15: Spearman correlation coefficients computed for Bangkok are shown. The coefficient calculated using reported holiday schedules and estimated parameters are shown by the dashed line. The coefficient computed using a holiday schedule shifted forward by one biweek is shown by the solid line. The histogram shows the distribution of correlations that were computed between the reported holiday schedule and 1,000 randomly ordered vectors of our parameter estimates.

profiles providing an optimal fit to the data show complete introduction of the new susceptibles immediately following the major holiday at the start of the school year. This is consistent with the idea that susceptible children impact the observed measles dynamics when they enter the school population.

4.4 Discussion and Conclusions

The development of inference tools for infectious disease models remains an important challenge to better understand disease dynamics and develop more effective control strategies. This work has demonstrated the flexibility inherent in large-scale nonlinear programming techniques and the ability of these techniques to efficiently

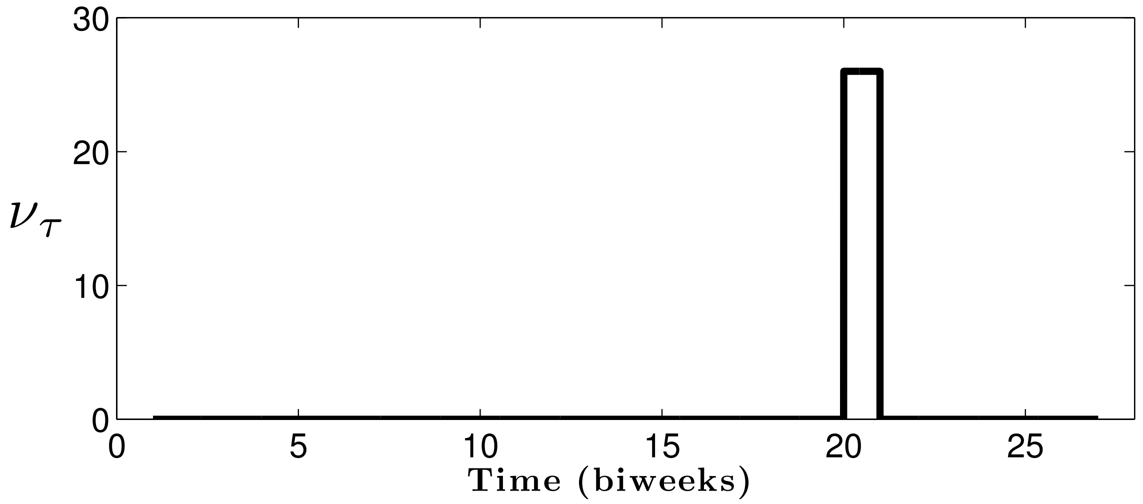

Figure 4.16: New York City estimates of seasonal weight on births, ν_τ.

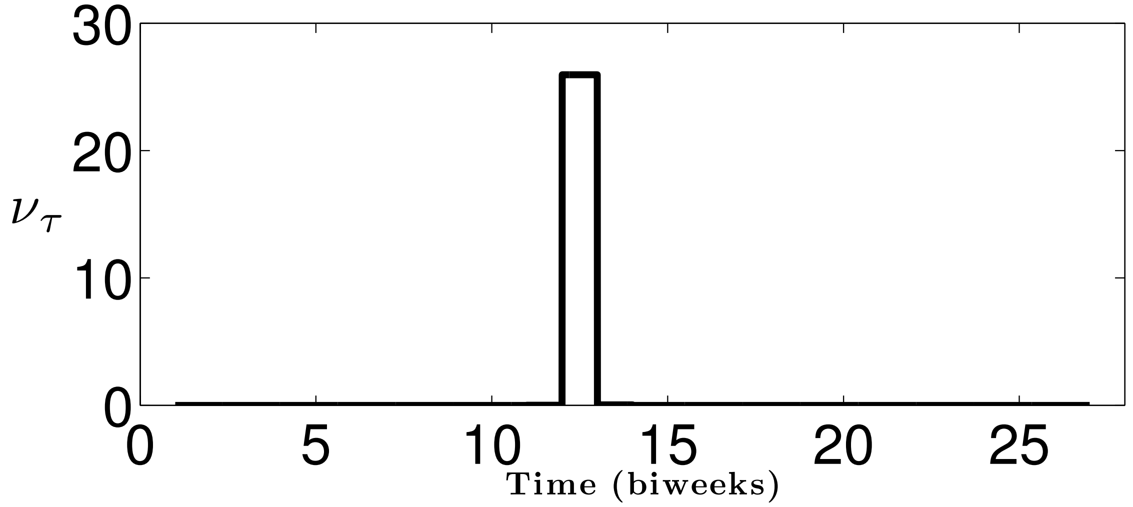

Figure 4.17: Bangkok estimates of seasonal weight on births, ν_τ.

estimate transmission parameters in multiple disease models using measles case count data. We demonstrated this efficiency and flexibility using three model formulations and four data sets. In all cases, including for time-series data sets of over 20 years, we were able to perform the estimations in less than 6 seconds. This computational efficiency and flexibility opens the door for investigating many alternative model formulations and encourages use of these techniques for estimation of larger, more complex time-discretized models like those with age-dependent dynamics, more complex compartment models, and spatially distributed data.

We validated our estimation approach by performing 10,000 estimations using simulated case data, and to demonstrate the flexibility of our approach, we presented estimations using measles case data from 3 different models. The first model we presented used a seasonally varying transmission parameter β. We first validated our approach by estimating seasonal transmission parameters using both simulated data and real measles data from London. We performed 10,000 estimations on simulated data (with different noise realizations) and showed that the approach was able to effectively recover the true seasonal transmission parameter. Furthermore, this approach estimates seasonal transmission parameters for London that are consistent with other estimates in the literature (He et al., 2010; Hooker et al., 2011).

Using real measles case data from both New York City and Bangkok, we estimated using 3 formulations, SVTP, SVEP, and SVIS, considering seasonality in the transmission parameter, the exponential parameter, and the introduction of new susceptibles. In all cases, the estimated seasonality showed correlation with school schedules. This is especially important given that the school schedules differ significantly for these two locations. The profile estimated using seasonal α's was practically identical to that estimated using seasonal β's. This result might not be too surprising, but this does highlight that care must be taken when relating the estimated seasonality to particular system phenomena (e.g., contact rate).

Perhaps more interesting are the estimation results for the model with a seasonal weighting of the births. Here, instead of assuming that new susceptibles always entered the population uniformly throughout the year, the model was formulated so that susceptibles could enter the population in any seasonal pattern. These estimation results show that all new susceptibles were introduced to the population immediately following long school holidays to best capture the dynamics observed in reported measles cases. Since all births clearly do not actually occur at this time, this result is consistent with the idea that the susceptible children impact observed measles dynamics when they enter the school population.

	NYC Measles			Bangkok Measles		
	Est	Low	High	Est	Low	High
α	0.9468	0.9405	0.9530	1.0137	1.0056	1.1062
γ_0	0.07363	0.06599	0.08188	0.01187	0.01013	0.01379
γ_{inc}	9.57E-5	6.30E-5	1.29E-4	1.00E-4	8.30E-5	1.17E-4
I_0	11360	8750	14810	6810	5120	9080
S_0	414800	393600	435900	352200	339500	362400
β_1	34.28	25.74	45.60	12.80	8.86	18.40
β_2	31.66	23.79	42.07	9.58	6.60	13.82
β_3	33.14	24.95	43.94	10.41	7.16	15.05
β_4	34.22	25.83	45.25	10.37	7.11	15.04
β_5	34.59	26.17	45.63	10.22	6.99	14.86
β_6	30.33	22.94	40.05	7.51	5.14	10.94
β_7	32.29	24.41	42.68	7.94	5.44	11.55
β_8	30.70	23.11	40.77	7.16	4.92	10.40
β_9	32.79	24.52	43.89	7.01	4.83	10.17
β_{10}	29.85	22.12	40.38	9.29	6.40	13.47
β_{11}	30.61	22.48	41.88	9.82	6.77	14.23
β_{12}	23.54	17.32	32.18	10.86	7.49	15.74
β_{13}	17.42	12.90	23.62	11.91	8.21	17.26
β_{14}	19.44	14.42	26.26	11.87	8.18	17.20
β_{15}	13.96	10.40	18.77	12.95	8.93	18.78
β_{16}	17.69	13.21	23.71	12.09	8.34	17.53
β_{17}	16.88	12.62	22.59	10.37	7.16	15.01
β_{18}	18.07	13.53	24.14	10.77	7.44	15.58
β_{19}	28.07	21.04	37.46	12.36	8.54	17.88
β_{20}	26.98	20.23	36.01	12.44	8.60	17.98
β_{21}	29.77	22.32	39.72	9.37	6.49	13.52
β_{22}	30.52	22.88	40.72	9.09	6.29	13.11
β_{23}	32.22	24.15	42.99	9.48	6.56	13.67
β_{24}	35.70	26.76	47.63	9.00	6.22	12.99
β_{25}	33.84	25.36	45.14	10.74	7.42	15.50
β_{26}	37.74	28.37	50.17	18.39	12.75	26.43

Table 4.2: Estimated parameters with 95% confidence intervals for measles in New York City and Bangkok using a seasonal transmission parameter.

5. ESTIMATING SEASONALITY WITH A CONTINUOUS-TIME INFECTIOUS DISEASE MODEL[*]

Mathematical models can enhance our understanding of childhood infectious disease dynamics, but these models depend on appropriate parameter values that are often unknown and must be estimated from disease case data. In this section, we develop a framework for efficient estimation of childhood infectious disease models with seasonal transmission parameters using continuous differential equations containing model and measurement noise. The problem is formulated using the simultaneous approach where all state variables are discretized, and the discretized differential equations are included as constraints, giving a large-scale algebraic nonlinear programming problem that is solved using a nonlinear primal-dual interior-point solver. The technique is demonstrated using measles case data from three different locations having different school holiday schedules, and our estimates of the seasonality of the transmission parameter show strong correlation to school term holidays. Our approach gives dramatic efficiency gains, showing a 40-400 fold reduction in solution time over other published methods. While our approach has an increased susceptibility to bias over techniques that integrate over the entire unknown state-space, a detailed simulation study shows no evidence of bias. Furthermore, the computational efficiency of our approach allows for investigation of a large model space compared to more computationally intensive approaches.

In this section, we use an SIR compartment modeling framework to develop a continuous-time model that includes both model and measurement noise. This

[*]Part of this section is reprinted with permission from "A Nonlinear Programming Approach for Estimation of Transmission Parameters in Childhood Infectious Disease Using a Continuous Time Model" by Word, D.P., Cummings, D.A.T., Burke, D.S., Iamsirithaworn, S., and Laird, C.D., 2012. Journal of the Royal Society Interface, Copyright 2012 by The Royal Society.

is then used to estimate model parameters and seasonal transmission profiles from measles case count data. It is assumed the individuals enter the susceptible compartment S when born, move to the infected compartment I upon acquiring the infection, and progress to the recovered compartment R upon recovery from the infection. After recovery from measles, individuals are assumed to attain lifelong immunity from the disease so there is no movement of individuals from the recovered compartment to the susceptible compartment (Anderson and May, 1991). The infection transmission assumes frequency dependence (Hethcote, 2000), and the transmission parameter is assumed to be seasonal with a periodicity of one year. The assumption of frequency dependent transmission is not reasonable for all diseases. While it may make intuitive sense that in larger populations one would have more contacts in a day, for childhood diseases like measles this is often not the case. Given that most children are in school systems with similar school structures, we assume they have a consistent number of contacts per day regardless of city size. This assumption is consistent with recent findings for measles in the UK (Bjornstad et al., 2002).

5.1 Problem Formulation

The differential equations describing the continuous-time seasonal SIR model are,

$$\frac{dS}{dt} = \frac{-\beta(y(t))S(t)I(t)}{N(t)} \cdot \varepsilon_M(t) + B(t) \tag{5.1}$$

$$\frac{dI}{dt} = \frac{\beta(y(t))S(t)I(t)}{N(t)} \cdot \varepsilon_M(t) - \gamma I(t) \tag{5.2}$$

where S is the number of susceptibles, I is the number of infectives, N is the total population, and $\beta(t)$ is the time-varying transmission parameter. The function $y(t)$ maps the overall horizon time into the elapsed time within the current year making $\beta(y(t))$ a seasonal transmission parameter with periodicity of one year. Births into the population, B, and the population, N, are known time-varying system inputs, and the recovery rate ($\gamma = \frac{1}{14\text{ day}}$) is a known scalar input. The variable ε_M represents multiplicative model noise which is assumed to be log-normally distributed with a mean of 1. Additive noise was investigated, but in those cases, the estimated values for the model noise showed an obvious temporal correlation with the data, indicating an unlikely model structure. However, the technique that we use can easily be modified to support different assumptions on the stochastic noise. Note that while this distribution can be an acceptable approximation for large data sets where the number of cases never nears zero, for small data sets this distribution becomes invalid since it does not allow for zero cases. In addition, since the reported number of cases must always be non-negative, the assumption of normally distributed measurement noise is only a reasonable approximation when the reported number of cases is not near zero. Given the size of the cities examined in this work and the number of reported cases over the given time horizons in the data sets, we find these assumptions acceptable here.

It is important to distinguish between incidence and prevalence in this problem formulation. The case count data available is the incidence, or the number of new cases reported over a given time interval. The model contains a state variable $I(t)$ that is the prevalence of the disease, or the number of cases present at a given point in time. In a given reporting interval, integrating over the number of new cases gives the incidence,

$$\text{Incidence} = \int_{t_{i-1}}^{t_i} \frac{\beta(y(\lambda))S(\lambda)I(\lambda)}{N(\lambda)} \cdot \varepsilon_M(\lambda)\, d\lambda. \tag{5.3}$$

To account for the difference in incidence and prevalence, a new state variable is introduced into the system through the following differential equation,

$$\frac{dQ}{dt} = \frac{\beta(y(t))S(t)I(t)}{N(t)} \cdot \varepsilon_M(t). \tag{5.4}$$

Here, $Q(t)$ represents the cumulative incidence at time t and provides a state variable that can be used to evaluate the incidence over a particular interval.

Not every individual that becomes infected is reported which leads to case counts being under-reported. While there are various methods to estimate the reporting fraction $\eta(t)$, in this work we use a straightforward approach to estimate a linearly varying reporting fraction that is similar to the susceptible reconstruction approach described in Finkenstädt and Grenfell (Finkenstädt and Grenfell, 2000). At the start of the estimation horizon we assume that the number of cumulative cases, C_0, and cumulative births, Y_0, are unknown. Prior to widespread vaccination almost every individual eventually contracted the disease. Therefore, on average the cumulative number of new cases should equal the cumulative number of new births. For a constant reporting fraction the cumulative cases and births are given by,

$$Y_t = \sum_{i=1}^{t} B_i + Y_0 \tag{5.5}$$

$$C_t = \sum_{i=1}^{t} \frac{R_i}{\eta} + C_0 \tag{5.6}$$

where B_i is the reported number of births at time i, Y_t is the cumulative number of births at time t, R_i is the reported number of cases at time i, and C_t is the cumulative number of cases at time t. We then minimize the sum squared error between Y_t and C_t to estimate the reporting fraction η. For the case of our estimations with data from London and Bangkok, we extend this basic formulation to estimate a reporting fraction that varies linearly in time.

In our problem formulation, the estimated reporting fraction is treated as a known input for the estimation of the disease model parameters. To obtain a fit, the estimation formulation requires that we minimize some measure of the model and/or measurement noise subject to the infectious disease model described by the differential equations given in Equations (5.1), (5.2), and (5.4).

As described in Section 3, we use the simultaneous approach for the solution of this dynamic parameter estimation problem. Here, we use collocation on finite elements to discretize the states into finite elements with fixed stepsize across the entire time horizon (Zavala et al., 2008). This converts the continuous differential equation model into an algebraic model that can be formulated as a nonlinear programming problem. A fifth-degree Gauss-Lobatto collocation technique is used to discretize the dynamics within these finite elements (Herman and Conway, 1996), and the discretized equations are included as equality constraints in the optimization problem. The effect of the instantaneous model noise, $\varepsilon_M(t)$, on the system is approximated by introducing an unknown noise term between each of the finite elements.

In our formulation, we include model noise between these finite elements so that, while inside a finite element Equations 3.7 - 3.10 are exact, between finite elements there can be state discontinuities. With model noise between finite elements, there is little benefit in the use of a higher order method. However, we initialize our problem using the deterministic case where no model noise is present (model noise terms are fixed to zero), and for this case this discretization is a high-order method.

Our estimation formulation becomes,

$$
\begin{aligned}
\min \ & \omega_M \sum_{i \in \mathcal{F}} (\ln(\varepsilon_{M_i}))^2 + \omega_Q \sum_{k \in \mathcal{T}} (\varepsilon_{Q_k})^2 \\
\text{s.t.} \\
& \frac{dS}{dt} = \frac{-\beta(y(t))S(t)I(t)}{N(t)} \cdot \varepsilon_M(t) + B(t) \\
& \frac{dI}{dt} = \frac{\beta(y(t))S(t)I(t)}{N(t)} \cdot \varepsilon_M(t) - \gamma I(t) \\
& \frac{dQ}{dt} = \frac{\beta(y(t))S(t)I(t)}{N(t)} \cdot \varepsilon_M(t) \\
& R_k^\star = \eta_k(Q_{i,k} - Q_{i,k-1}) + \varepsilon_{Q_k} \\
& \bar{S} = \frac{\sum_{i \in \mathcal{F}} S_i}{\text{len}(\mathcal{F})} \\
& \bar{\beta} = \frac{\sum_{i \in \tau} \beta_i}{\text{len}(\tau)} \\
& 0 \leq I(t), S(t) \leq N(t) \\
& 0 \leq \beta(y(t)), Q(t)
\end{aligned}
\tag{5.7}
$$

where the differential equations are discretized using equations (3.7-3.10) and are shown here in their differential form for simplicity. The index k is a time point within the set of reporting times, $\mathcal{T}$, while $\mathcal{F}$ is the set of all finite elements used in the discretization. The reporting fraction η_k accounts for under-reporting over a given time interval spanning $k-1$ to k, $R_k^\star$ is the actual reported incidence over a given time interval, ε_{Q_k} is the measurement noise, and ω_M and ω_Q are weights for the noise terms. Based on the assumption of normality in ε_{Q_k} and $\ln(\varepsilon_{M_i})$, the ratio of these weights should be equal to the inverse ratio of the variance in the estimated noise terms. Since the variances are not known a priori, a simple bisection approach is used to solve for the ratio of weights until $\frac{\omega_M}{\omega_Q} = \frac{\sigma_Q^2}{\sigma_M^2}$ where σ_Q^2 is the calculated variance of the estimated noise terms ε_{Q_k}, and σ_M^2 is the calculated variance of the estimated noise terms $\ln(\varepsilon_{M_i})$ (using standard mean squares). The estimation formulation is solved completely for each iteration of the bisection method. These weights determine the tradeoff between model and measurement noise in the objective function. This straightforward approach was used since sensitivity studies (presented later in the paper) show that the estimates are relatively insensitive to the selection of these weights. However, other techniques have been proposed for determining these weights and could also be used (Varziri et al., 2008; Hooker et al., 2011). Two additional variables are added to the problem formulation for use in calculating confidence regions. Here, $\bar{S}$ is the average population of susceptibles over the time horizon and $\bar{\beta}$ is the average value of β across the yearly set of discretizations τ.

It is important to point out that the objective function used in this formulation is the extended log-likelihood and only an approximation of the true log-likelihood for the parameters. The use of this likelihood for estimating parameters can cause considerable bias in both the parameters and their uncertainty (Lee et al., 2006). In practice bias may not always be observed, but care should be taken when evaluating results from this approach. The simulation study discussed in Section 5.2.1 shows no significant bias. However, if bias is a concern then this efficient formulation and approach can still be used to initialize an unbiased approach.

This research focuses on the estimation of the seasonal transmission profile $\beta(y(t))$. While the case data shows strong seasonality, the functional form of the transmission profile is unknown so it is undesirable to force β to take the form of a particular periodic function (e.g. a sine function). Therefore, we discretize the transmission profile along finite element boundaries, assuming a constant value through the finite element. In previous work, $\beta(y(t))$ was further restricted using total variation regularization (Word et al., 2010). However, in this work it was found that regularization was unnecessary when the discretization of β matched the reporting interval of the case data. Some form of regularization or restriction of β would be necessary if β is discretized more finely than the reporting interval.

5.2 Estimation Results

To demonstrate the effectiveness of our approach, we first estimate parameters using simulated data from an SIR model. We then perform estimation using three real data sets from different settings. In our estimations we use existing measles case count data for London (Bjornstad et al., 2002), New York City (Yorke and London, 1973), and Bangkok, which has been made available to us by the Thailand Ministry of Public Health (of Epidemiology, 1986). These data are shown in Figures 5.1, 5.2, and 5.3 respectively. These data sets also include yearly birth records and populations. The London data set was chosen since it has been widely studied and provides a comparison of our model results with literature results. The New York City and Bangkok data sets are from cities with very different social settings and entirely different school holiday schedules. This allows us to compare estimated transmission profiles on locations with different school term schedules. The New York City data contain monthly reported case counts. The Bangkok data include monthly case counts and annual age distributions. There is regular active surveillance coupled to the passive surveillance in order to assess the performance of the passive surveillance system. The data is fully anonymized and laboratory confirmation is reported when available. The populations are assumed to vary linearly throughout the year, and the birth rates are assumed to be uniform throughout the year.

To perform these estimations, we first format the data as required by the AMPL input data file format. We formulated the model shown in Equation 5.7 with discretized differential equations using the algebraic modeling language AMPL (Fourer

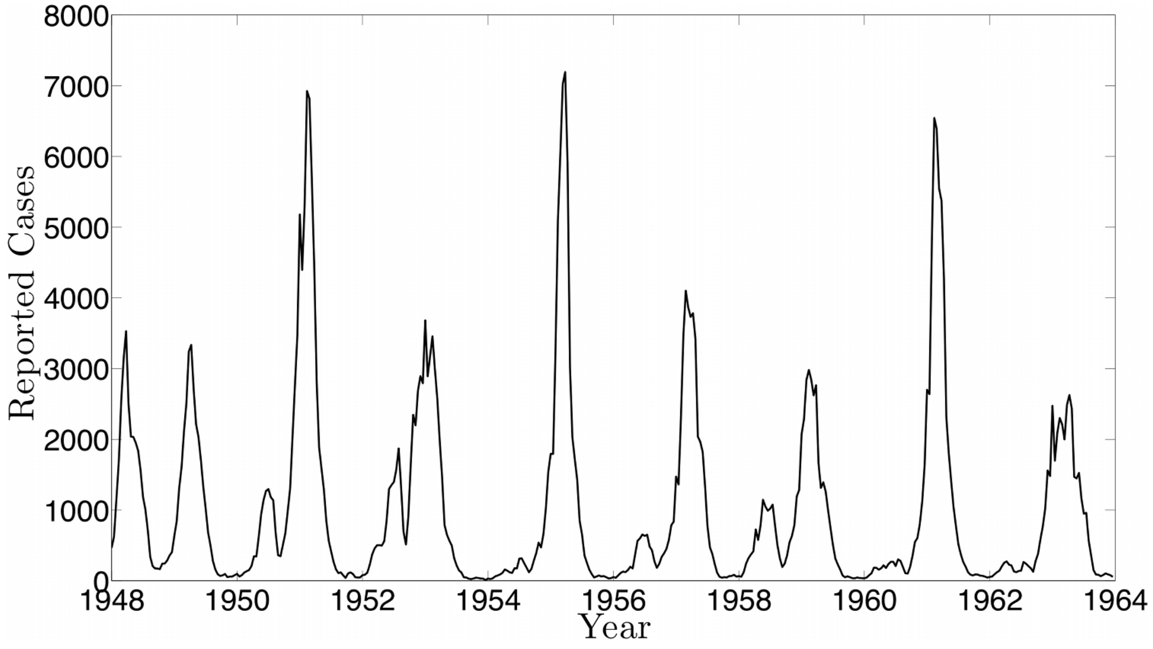

Figure 5.1: The reported number of cases for London.

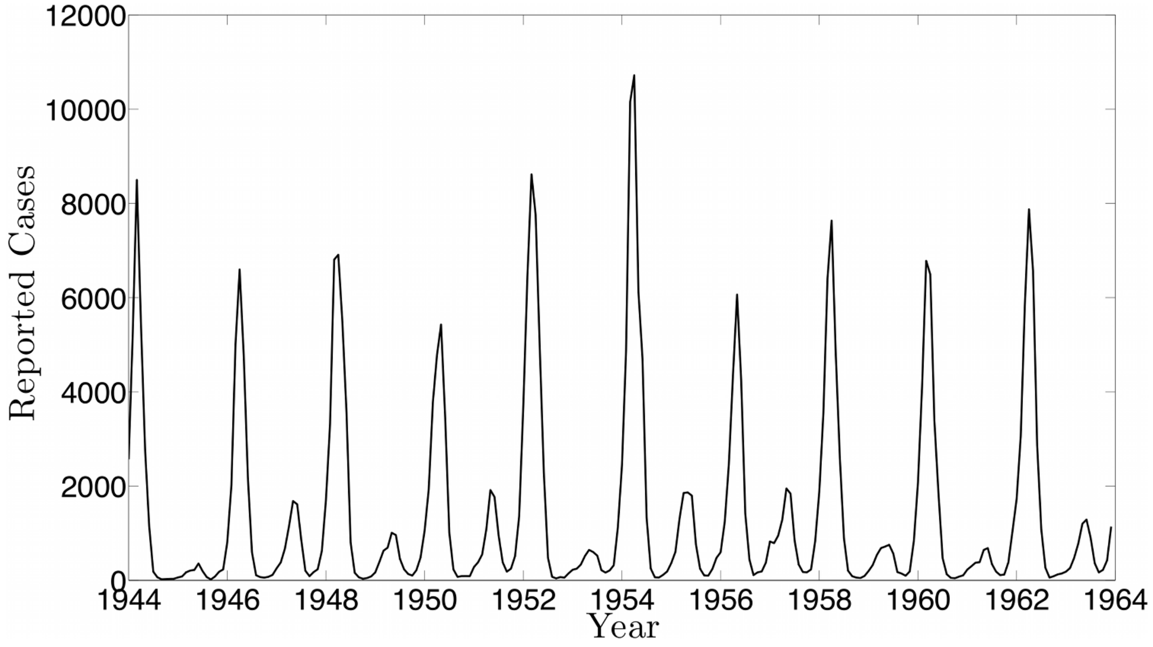

Figure 5.2: The reported number of cases for New York City.

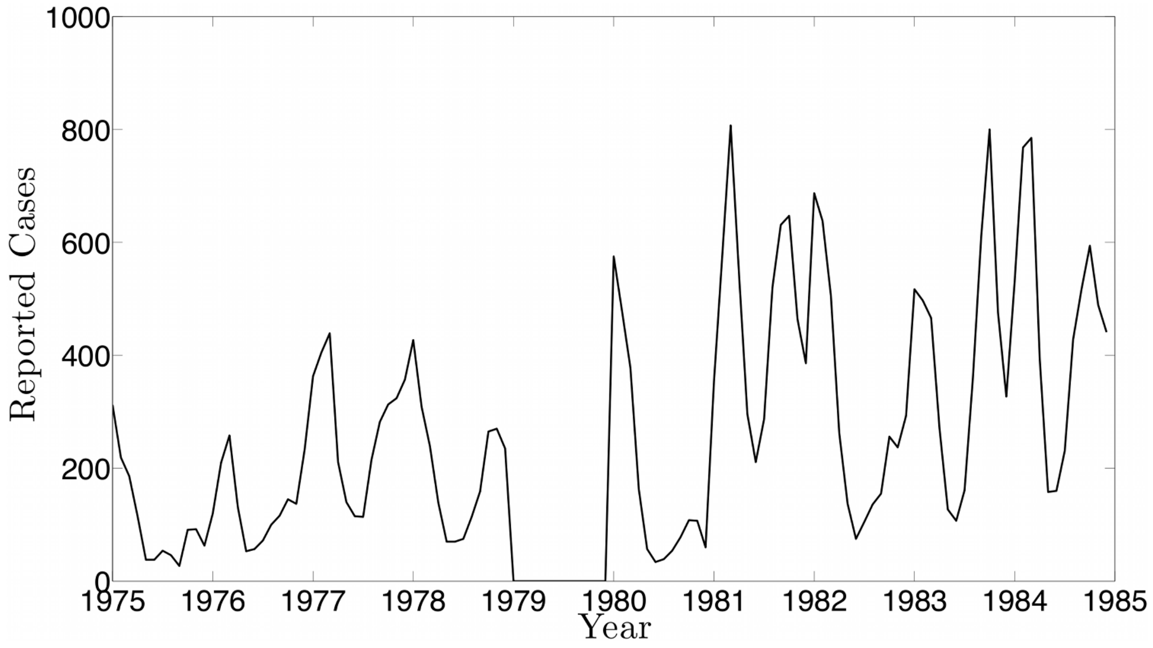

Figure 5.3: The reported number of cases for Bangkok.

et al., 1993). Any modeling language coupled with a reliable large-scale nonlinear optimization solver could be used. We solve the problem using the open-source nonlinear solver Ipopt (Wächter and Biegler, 2006). The weights for the objective function are found using the iterative process discussed previously. These weights are then fixed to find the confidence intervals and confidence regions.

Effective initialization is important for successful solution of general non-convex nonlinear programming problems. Here, all problems were initialized simply by setting all $S_{i,j} = 1 \cdot 10^5$, $I_{i,j} = 1 \cdot 10^2$, and $Q_{i,j} = 0 \; \forall \; i \in \mathcal{F}$ and j collocation points, and $\beta_i = 1 \; \forall \; i \in \tau$. Here, $\mathcal{F}$ is the set of finite elements and $\mathcal{B}$ is the set of discretizations of β. While this is a very crude initialization, the formulation is robust, and we first solve the deterministic problem, fixing the model noise terms to zero, before solving the formulation with model noise terms included.

Our estimates showed a strong correlation between $\bar{S}$ and $\bar{\beta}$, and the quality of our fit to the data differs dramatically outside of a narrow range of values. Confidence regions, derived from the likelihood ratio test (Seber and Wild, 1989; Gallant, 1987), are constructed as described in Rooney and Biegler (2004) to show the region in which these values could be expected to lie. The regions are constructed over pairs of parameters by fixing the two parameters and reoptimizing over the remaining variables. In addition, likelihood ratios are used to construct confidence intervals for β by fixing each β_i independently and allowing optimization over all other β_i's. These confidence regions and intervals were done using the extended likelihood which is only an approximation of the true likelihood. The simulation study performed in Section 5.2.1 indicates that this approximation still provides reasonable confidence intervals. In the simulation study, the estimated confidence intervals are slightly more conservative than what the simulations would suggest as necessary. Appendix G contains the AMPL code used to generate these confidence regions.

5.2.1 Simulation

In order to test the estimation procedure using known parameter values, we perform stochastic simulations with an SIR model. The simulations were performed using MATLAB. Our simulations used a constant population of 10,002,000, a recovery rate of $\frac{1}{14\,\mathrm{day}}$, a birth rate of 2.95% of the population per year, and a reporting fraction of 1. To generate 20 years of case data, the deterministic model was integrated for 100 years to achieve a cyclic steady state. The final values from this simulation were used as the initial values for the stochastic simulation. The simulations were performed with the same model as that used for the estimation except that the timestep used within the Matlab integration routine was a half day. Model noise was drawn from a log-normal distribution with mean 1 and $\sigma = 0.05$. Measurement noise was applied to the reported cases and was drawn from a normal distribution with mean 0 and a standard deviation of 1,000. While the assumption of normally distributed noise is not valid for data sets with a low number of reported cases, we assume it to be a reasonable distribution for our simulations since in 10,000 simulations the reported number of cases never fell below 2,000.

The simulation was run 10,000 times with a reporting fraction of 1 to generate simulated case data, and estimations were run on each of these simulations using 12 finite elements per year. Figure 5.4 demonstrates that our estimation approach gives an extremely good estimate for β using data from the SIR simulations. The solid (red) line shows the true parameter values used for all 10,000 simulations. The (blue) circles show the mean of the estimated values for the parameters as well as the 2.5 and 97.5 quantiles for the parameters estimated from all the simulations (giving the 95% confidence intervals for these estimates). The (black) triangles show the estimated parameters from an estimation on a single randomly selected simulated data set, and the dashed (black) lines show the 95% confidence intervals generated for this single estimation using the likelihood ratio test. The true parameter values are included well inside these confidence intervals. The confidence intervals calculated from the likelihood ratio test give more conservative intervals than what the 10,000 simulations would suggest are necessary and actually cover over 97% of the values estimated from all 10,000 simulations. This is not unexpected given that the likelihood ratio test confidence intervals are profile likelihoods, determined by fixing only one parameter at a time, allowing the other parameters to be optimized.

Furthermore, the mean values of the estimated parameters over all 10,000 simulations agree with the true values used in the simulation. There is no significant bias observed in the estimate of the seasonal transmission parameters for the simulated data used in this study.

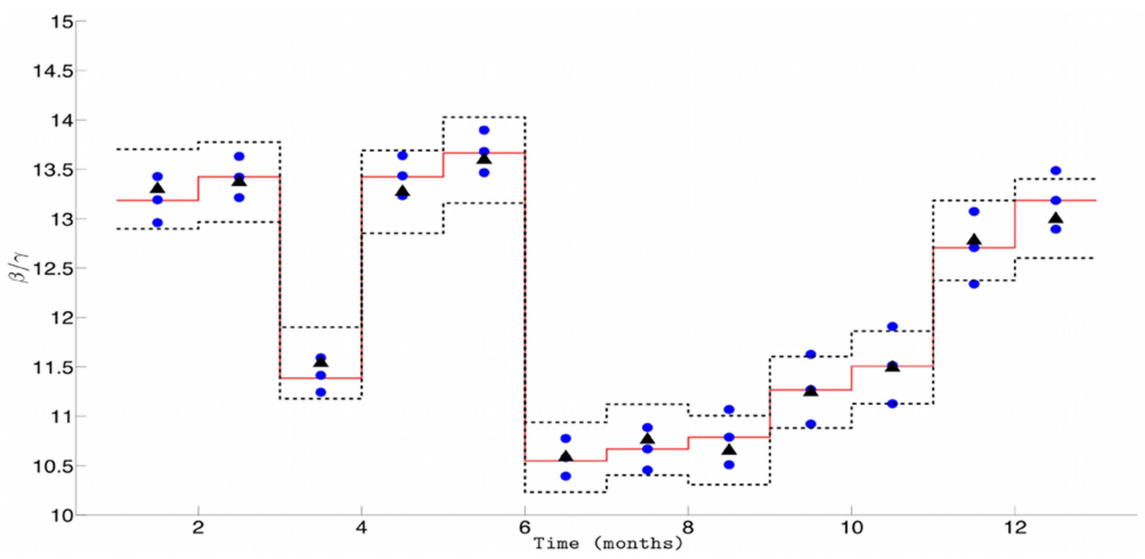

Figure 5.4: The estimated transmission profile β (black triangles) for a single data set with 95% confidence intervals (– –) found using likelihoods. The true values of β used in the stochastic SIR simulation (solid red line). The 2.5th, 50th, and 97.5th quantiles of the estimates from the simulation (blue circles).

5.2.2 London

The time horizon studied for London was 1948-1964. The reporting fraction estimated using our approach varies linearly from 50.65% to 42.55% over the time horizon studied. This data set originally had case counts reported weekly, however the data set used here has combined the data so that case counts are given biweekly. This aggregated data was also used by Finkenstädt and Grenfell (2000). The estimation for London was performed using 26 finite elements per year, and the transmission profile was discretized so that each β_i contained 1 finite element, giving 26 discretizations in β. The normalized estimated seasonal transmission profile for London is shown in Figure 5.5. This figure also compares this result with the findings from Finkenstädt and Grenfell (2000) and Cauchemez and Ferguson (2008). The time horizon used in this work and by Cauchemez and Ferguson was the years 1948-1964 (Cauchemez and Ferguson, 2008), but the time horizon used by Finkenstädt and Grenfell was the years 1944-1964 (Finkenstädt and Grenfell, 2000). Despite the differing time horizons, it is clear that the seasonal pattern estimated from our approach is similar to these other results. The estimated profile appears to be correlated with school term holidays with the transmission profile decreasing during the school breaks that occur during the Easter holiday around biweek 8 and the summer holiday over biweeks 15-18. The Christmas holiday occurs at biweek 25, but there is no immediate effect captured in our estimates. The lack of any immediate effect could be due to delays in reporting over the Christmas holidays (Fine and Clarkson, 1982).

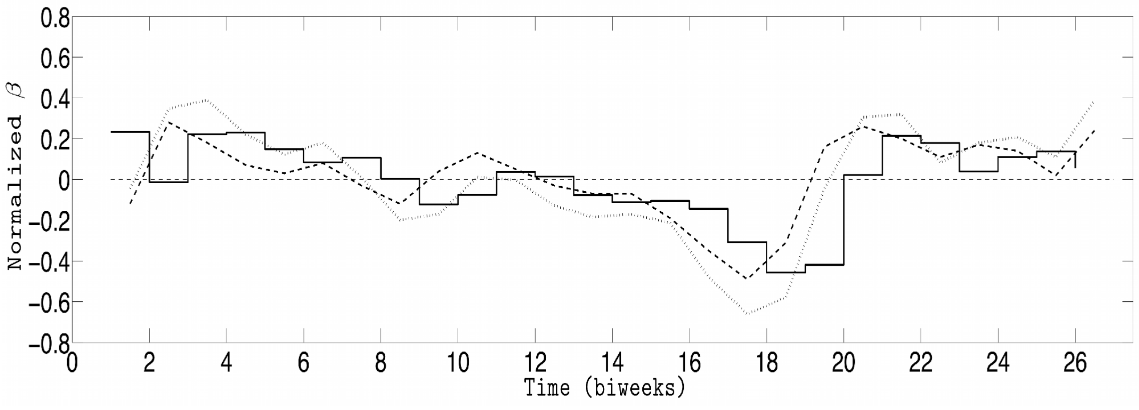

Figure 5.5: The transmission profile estimated for London by Finkenstädt and Grenfell (2000) (– –), Cauchemez and Ferguson (2008) (· · ·), and this work (—).

Figure 5.6 shows the non-normalized pattern for the estimated seasonal transmission profile. The estimated seasonal transmission parameter $\beta(t)$ is on a per day basis, and the mean estimated value is $\bar{\beta} = 0.95$. It does appear that our estimates may be shifted by one biweek relative to the school holidays, but this could be due to a slight delay in reporting.

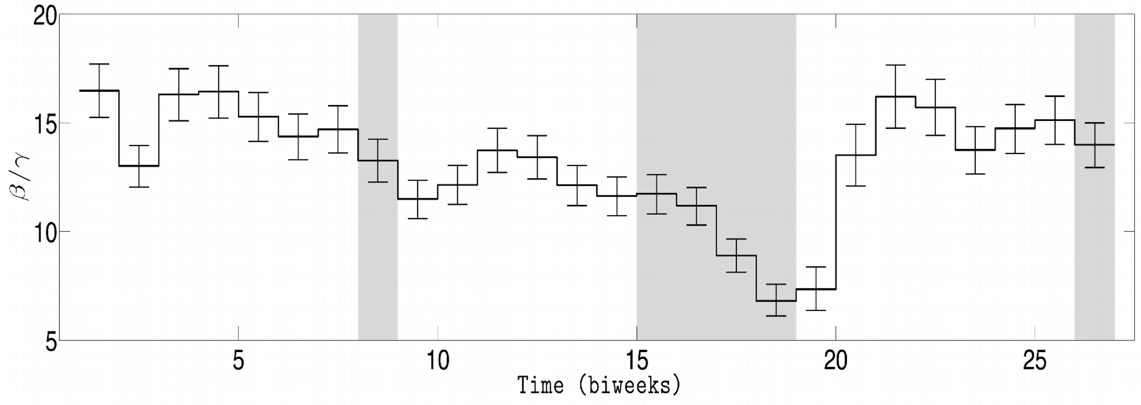

Figure 5.6: The estimated transmission profile β with confidence intervals. School term holidays are shaded.

In Figure 5.7, we show nonlinear confidence regions for the mean transmission parameter value $\bar{\beta}/\gamma$ against the mean susceptible fraction over the mean population, $\bar{S}/Population$, where the means were taken over the entire time horizon. As expected, this shows a dependence between the estimated values for $\bar{\beta}$ and $\bar{S}$. The shape of these confidence regions provide some insight into why it may be difficult to accurately estimate the absolute magnitude of the transmission parameter. An increase in $\bar{\beta}$ can be offset by a decrease in $\bar{S}$ with little change in the objective function value. However, even though there is a difference in the absolute magnitude of the transmission parameter, we see little difference in the seasonal pattern estimated for the points within the indicated confidence region.

5.2.3 New York City

The estimation for New York City was performed using monthly reported data from 1944-1963. The reporting fraction estimated using our approach was almost constant around 11% throughout the time horizon studied, and the reporting fraction was assumed constant in our estimates. The discretization strategy used 12 finite elements per year, and β is again assumed to be constant within each finite element.

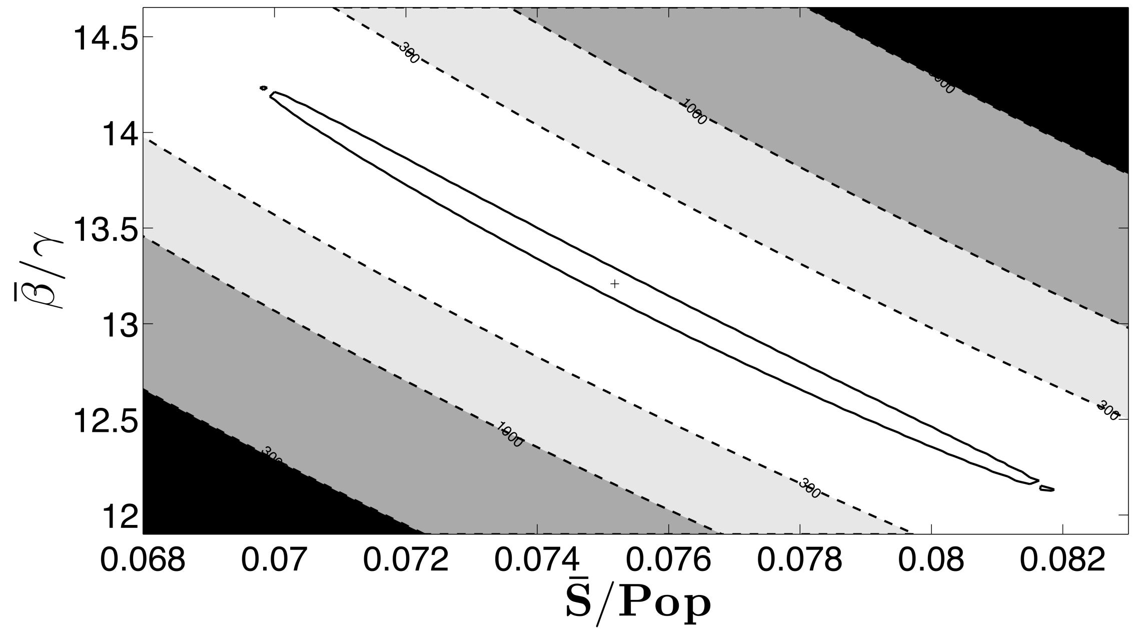

Figure 5.7: A contour plot from estimations using London data showing the contours of the left hand side of the χ^2 test coming from the likelihood ratios (– –) and the 95% confidence region (—) for $\bar{\beta}$ and $\bar{S}$ divided by the population. Regions represent profile likelihood contours where likelihood ratios are calculated based on a re-optimization of all other parameters. The "+" indicates the optimal estimated value.

This gave 12 discretizations for the seasonal transmission profile β. The estimated seasonal transmission profile for New York City is shown in Figure 5.10. This profile also shows strong correlation with the summer school term holiday that the New York City Board of Education reported occurring from mid-June through mid-September, or over the finite elements of approximately 6.5-9.5. There are school holidays around the end of the year, however, these holidays are much shorter than the reporting interval of the data.

Figure 5.8 shows the 95% confidence region for $\bar{\beta}$ and $\bar{S}$. The optimal estimated $\bar{S}$ was 11.1% of the mean population, and the optimal estimated $\bar{\beta}$ was 0.65. Our approach successfully estimates a seasonal transmission pattern that shows strong

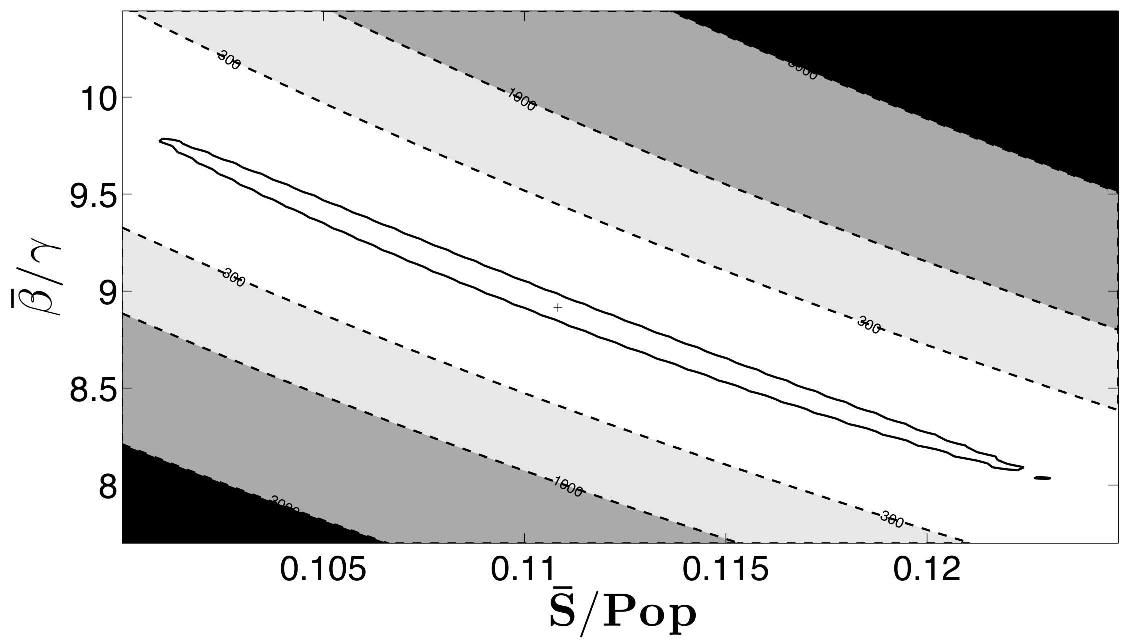

Figure 5.8: A contour plot from estimations using New York City data showing the contours of the left hand side of the χ^2 test coming from the likelihood ratios (– –) and the 95% confidence region (—) for $\bar{\beta}$ and $\bar{S}$ divided by the population. Regions represent profile likelihood contours where likelihood ratios are calculated based on a re-optimization of all other parameters. The "+" indicates the optimal estimated value.

correlation to school terms.

5.2.4 Bangkok

In addition to data obtained from locations with a single large summer break, we also performed estimates using measles data for Bangkok, Thailand. Thailand has two school term holidays – one in the spring and one in the fall. The estimation for Bangkok was performed using monthly reported case count data from the years 1975-1984. This data set contains significant underreporting with the estimated reporting fraction varying linearly from 1.1% at the start of the time-series to 4.5% at the end of the time series. In addition, case counts are missing for the year 1979.

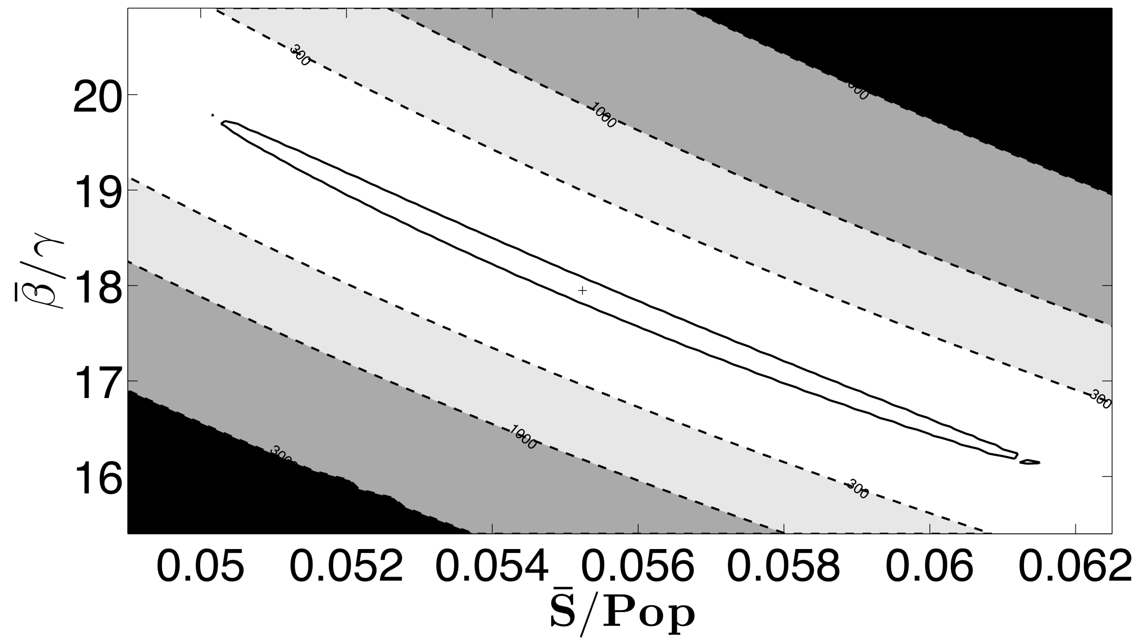

Figure 5.9: A contour plot from estimations using Bangkok data showing the contours of the left hand side of the χ^2 test coming from the likelihood ratios (– –) and the 95% confidence region (—) for $\bar{\beta}$ and $\bar{S}$ divided by the population. Regions represent profile likelihood contours where likelihood ratios are calculated based on a re-optimization of all other parameters. The "+" indicates the optimal estimated value.

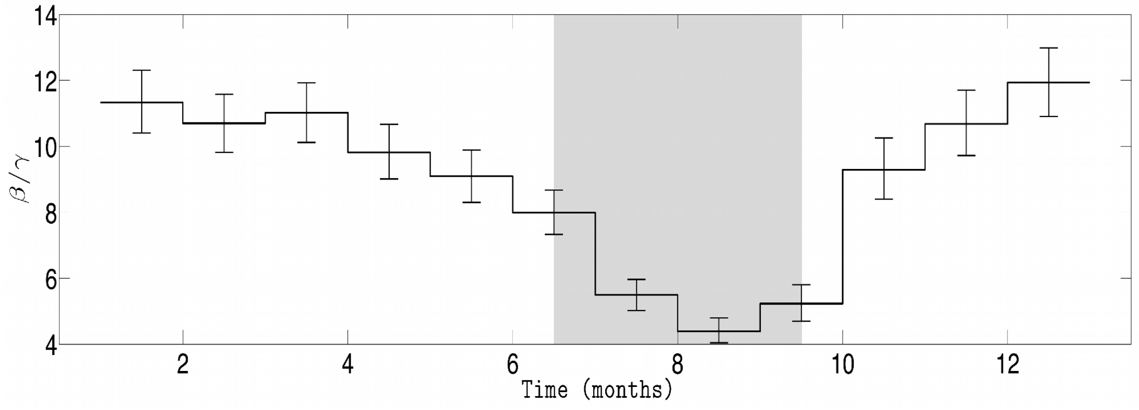

Figure 5.10: The estimated transmission profile β for New York City with shaded school term holidays.

The discretization strategy used 12 finite elements per year with β discretized by finite elements, giving 12 discretizations for β. Since no data is available for the year 1979, these points were excluded in the objective function so that they would not affect the estimation while still allowing the model to simulate the states through this year. The estimated seasonal transmission profile β for Bangkok is shown in Figure 5.11. This profile shows correlation with the two school term holidays that occur from the beginning of March until mid-May and the entirety of October, or over the finite elements of approximately 3-5 and 10-11. There is an obvious lag between our estimated drop in β and the start of the school holidays. This lag is likely due to a lag in the reporting of case data. There are consistently extra cases reported in January resulting from a backlog of reports that are not processed at the end of the year due to worker holidays. This suggests that cases are being reported as occurring when reports are processed rather than when the cases actually occurred which would cause a lag in the estimates.

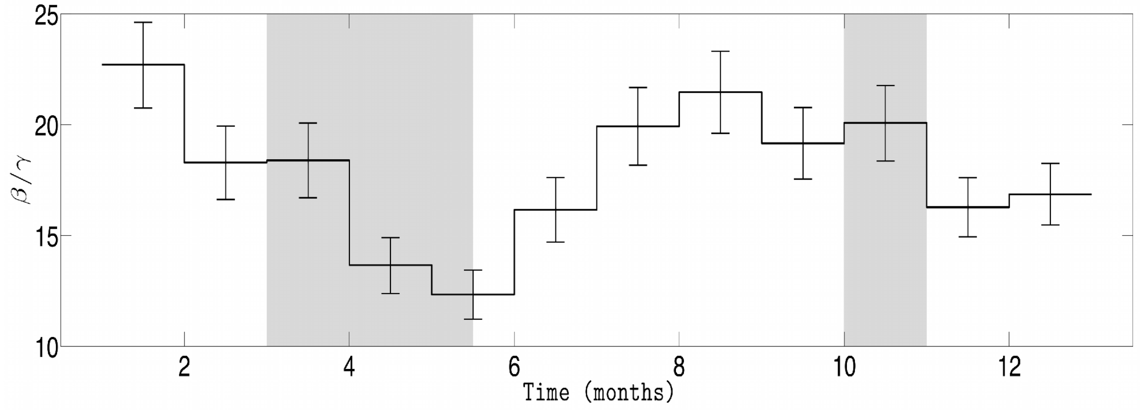

Figure 5.11: The estimated transmission profile β for Bangkok with shaded school term holidays.

Figure 5.9 shows the 95% confidence region for $\bar{\beta}$ and $\bar{S}$. The optimal estimated $\bar{S}$ was 5.5% of the mean population, and the optimal estimated $\bar{\beta}$ was 1.28.

5.2.5 Input Sensitivity Analysis

Our estimates are dependent upon the inputs we use in our model. We use recorded data for birth inputs and for population inputs, but no data is available for the reporting fraction and recovery rate, and we would like to know how sensitive our estimates are to the values used for these inputs. In addition, we use an iterative approach to set the weights in the objective function, and we would like to know how changing these weights will affect our estimates.

We first examined the effect of varying the reporting fraction on our estimates of β. To investigate this, we varied the value of the reporting fraction over a wide range about our estimated value. Using each new reporting fraction, we solved the same problem as before. Figure 5.12 shows the estimated average value of β for New York and the optimal objective function value as the reporting fraction changes. This plot shows that even for small changes in the reporting fraction away from the estimated value there is significant decrease in the average value of β. The objective function values show that the best data fit occurs when the reporting fraction is where we also get the highest $\bar{\beta}$, and this reporting fraction is the same as that estimated using the approach described in Section 5.1.

In Bangkok, there is a significant difference in the reporting fraction at the beginning and end of the time horizon studied and a time varying reporting fraction is needed. We use a linearly varying reporting fraction throughout the time horizon, and for our sensitivity study we keep the same slope as we estimated before. Figure 5.13 shows the estimated $\bar{\beta}$ and optimal objective function values as the initial value of the reporting fraction is varied. Since the reporting fraction estimated previously

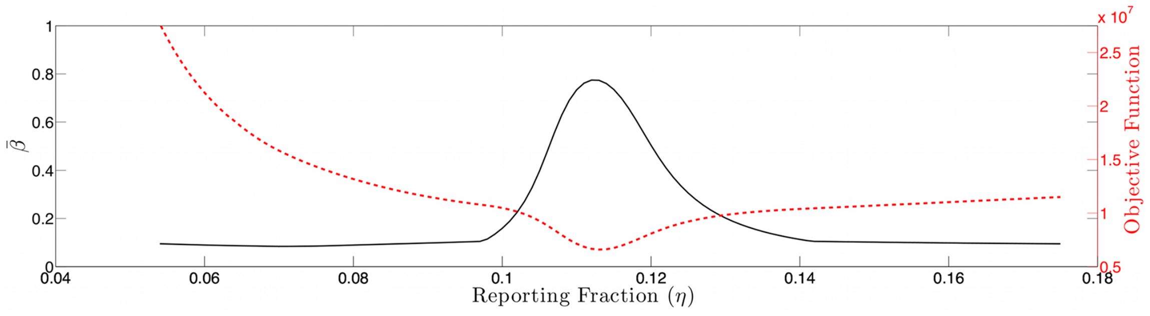

Figure 5.12: The $\bar{\beta}$ estimated for New York City is shown as a function of the reporting fraction (—). The objective function value is also shown as a function of the reporting fraction ($\cdots$).

is so low at the beginning of the time horizon and must be positive, we are unable to lower the initial value more than about half of a percent. Where we see the minimum in the optimal objective function value is also where we find the actual value of the reporting fraction that we estimated previously.

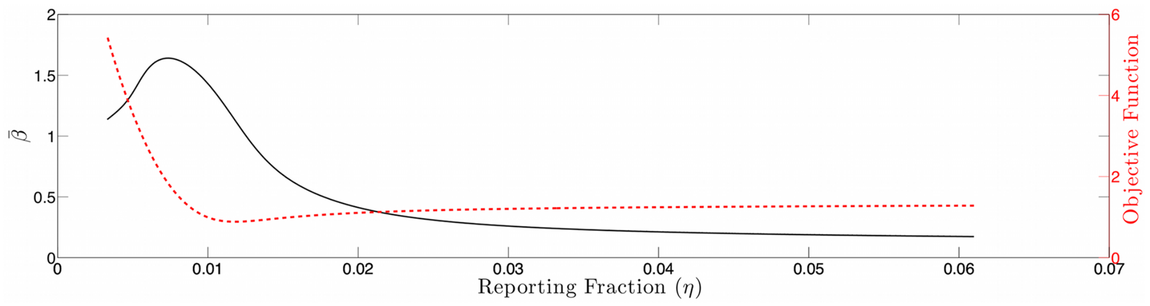

Figure 5.13: The $\bar{\beta}$ estimated for Bangkok is shown as a function of the initial value of the reporting fraction (—). The objective function value is also shown as a function of the reporting fraction ($\cdots$).

The recovery rate can also affect the dynamics of the system, and different sources use different values (typically $\frac{1}{13}$ or $\frac{1}{14}$). Figure 5.14 shows the change in the estimated $\bar{\beta}$'s for New York City and Bangkok as the recovery rates are varied. For both cities, the changes in $\bar{\beta}$ are not dramatic at any point, but simply change slowly throughout the range examined. This indicates that for reasonable values for the recovery rate our estimates are not dramatically affected.

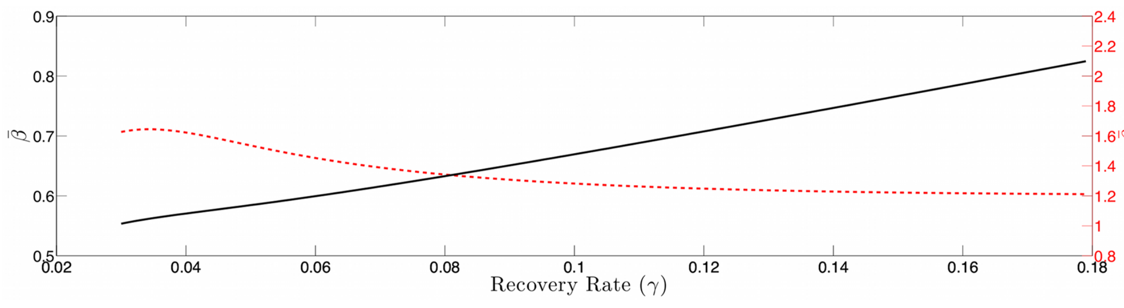

Figure 5.14: The $\bar{\beta}$'s estimated for Bangkok ($\cdots$) and New York City (—) are shown as functions of the recovery rate.

Finally, we also show the estimation results as a function of the ratio of the weights in the objective function. Here, we vary this ratio two orders of magnitude on either side of the value found using our iterative procedure. Figure 5.15 shows the value of $\bar{\beta}$ for both New York City and Bangkok as the weights are varied. The mean value of the transmission parameter changes little over a range of values near the estimated weights. Furthermore, the pattern exhibited by the estimated transmission parameter is nearly identical over this entire range.

5.3 Discussion and Conclusions

Successful estimation of parameters in dynamic models for childhood infectious diseases from time-series data presents several challenges. Typically, reported cases (the incidence) are the only available data, while there is little information about the susceptible population. Therefore, approaches must simultaneously estimate the

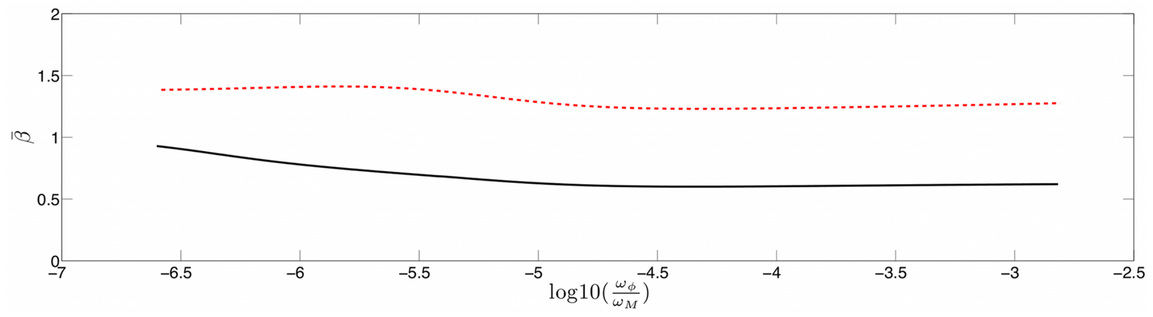

Figure 5.15: The $\bar{\beta}$'s estimated for Bangkok ($\cdots$) and New York City (—) are shown as functions of the log of the ratio of the weights on the noise terms in the objective function.

prevalence and the unknown susceptible states. Furthermore, the case data is often significantly under-reported, the reporting interval is often longer than the serial interval of the disease, and the models are highly nonlinear. This paper presented a nonlinear programming approach for estimating the unknown states and the seasonal transmission parameter using a continuous-time model with both measurement and model noise.

Continuous-time formulations offer several advantages over discrete-time formulations for estimation of infectious disease models. Data can be handled in its native form regardless of the reporting interval. This was demonstrated by using biweekly reported data from London and monthly reported data from New York City and Bangkok. Using data in its native form is a significant advantage for diseases with short serial intervals where it would be unreasonable to have data reported at the same interval.

The estimation approach outlined in this paper is highly efficient. The estimation formulation using continuous SIR models is a nonlinear optimization problem subject to differential equations as constraints. The use of the simultaneous or full-discretization approach produces a large-scale algebraic nonlinear programming

	Variables	Constraints	CPU Time (sec)
London	16459	15963	190
NYC	8929	8663	300
BKK	4477	4331	76

Table 5.1: Problem sizes and solution times of the London, New York City (NYC), and Bangkok (BKK) estimation problems studied in this paper. All problems were solved on a 3.0 GHz Intel Xeon processor and times are reported in seconds.

problem. Nevertheless, efficient solutions are possible since the simulation is not converged at each iteration. The solution times for all estimations are shown in Table 5.1. These are the full solution times, including the times required to initialize the problems and find the weights to be used in the objective functions. Significant reductions could be made by initializing the problem well and by giving good initial guesses for the objective function weights. Recent work by Hooker et al. solves a similar problem formulation in approximately 2 hours (Hooker et al., 2011), the MCMC estimation performed by Cauchemez and Ferguson required approximately 20 hours per run (Cauchemez and Ferguson, 2008), and the plug-and-play method of He et al. (He et al., 2010) requires approximately 5 hours. None of our estimations take longer than 5 minutes. The efficiency of this fully simultaneous approach opens the door to explore many more model structures efficiently and provides a framework that is scalable to large spatially distributed estimations.

Figures 5.7, 5.8, and 5.9 show a strong inverse correlation between the estimated $\bar{\beta}$ and $\bar{S}$ as seen by the narrow, elongated confidence regions. This result is not unexpected when compared with the approximate expression relating β to $\bar{S}$ given by $1/\bar{S} = \bar{\beta}/\gamma$ (Anderson and May, 1991). This relation gives a curve lying approximately through the middle of the 95% confidence region in Figures 5.7, 5.8, and 5.9.

These elongated regions indicate that the estimation is sensitive along this line and that care should be taken when interpreting absolute values for β or S. It should also be noted that the estimations produced nearly identical patterns in $\beta(t)$ within these confidence regions.

More importantly, several recent publications have reported estimated values of the seasonal transmission parameter, and corresponding R_0 values, that are higher than estimates provided in Anderson and May (1991). For example, the reported estimates of He et al. (2010) for London give an R_0 of 57 with 95% confidence intervals of 37 and 60. There is significant complexity in finding R_0 values while considering seasonal transmission rates, and it is difficult to compare results arising from different model structures. Using the approximate relationship $R_0{=}\bar{\beta}(t)/\gamma$, we estimate $R_0{=}13.3$ in London with 95% confidence intervals of 12.1 and 14.3. Our estimates for New York City ($R_0{=}9.1$) and Bangkok ($R_0{=}17.9$) also give values for R_0 that appear consistent with values reported for measles in Anderson and May for other cities (Anderson and May, 1991) and with values approximated using the average age of infection.

The estimated transmission profiles from all three cities show strong correlation with school holidays despite the very different holiday schedules seen between London, New York City, and Bangkok. For Bangkok and NYC there was a lag observed in the estimated transmission profiles that showed the drop in transmission as occurring after the holiday had begun. This is probably due to a lag in reporting causing cases to be reported well after their occurrence and the incubation period of measles causing cases to be observed after the start of the holiday even though the infection occurred before the holiday.

This overall approach for estimating continuous-time infectious disease models is reliable, flexible, and efficient. Although the use of extended-likelihood may not be guaranteed to provide unbiased estimates, the simulation results showed no evidence of bias. Solutions to the nonlinear programming problem were possible with a general initialization strategy, and effective parameter estimates are possible, even in the face of challenging sets of data that contain missing years, severe under-reporting, and significant noise. It is straightforward to switch between diseases with different serial intervals or data sets with different reporting intervals. The approach is independent of model specifics. For example, it would be straightforward to add additional compartments to the model, such as adding an E compartment to make an SEIR model that would account for individuals that have been exposed to a disease but cannot yet infect susceptibles. One could also add a compartment to account for portions of the population that were vaccinated against a disease. Furthermore, the approach is highly efficient, making it appropriate for much larger problem formulations, or for rapid exploration and comparison of multiple model structures.

6. PARALLEL PROGRESSIVE HEDGING FOR PARAMETER ESTIMATION*

While the focus of Sections 4 and 5 was on estimating transmission parameters for individual cities with large populations, a more interesting problem for public health officials is looking at a spatial model of disease spread. For accurate estimation of disease dynamics in small cities where fadeout is observed, models must consider the transmission of disease from large cities where the disease is endemic to small cities. Such spatial-temporal models can become prohibitively large for many solution approaches. To address problems of this scale parallel estimation algorithms must be developed.

In this section we present the progressive hedging algorithm for parallel estimation of transmission parameters in a very large-scale measles model that includes data from many cities. This approach decomposes the problem into multiple scenarios where each city contributes data for one scenario. We demonstrate the capability of this algorithm to efficiently compute parameter values using real-world observations.

6.1 Parameter Estimation and Stochastic Programming

Parameter estimation remains an essential component in the development of accurate, reliable models of dynamic systems. Nonlinear programming (NLP) has proven to be a powerful tool for parameter estimation, and many algorithms have been utilized with success (Betts, 2010; Aster et al., 2012; Zavala et al., 2008; Leibman et al., 1992; Tjoa and Biegler, 1991; Biegler and Zavala, 2009; Zavala and Biegler, 2006). However, problem size and complexity continues to increase, driving the demand for more powerful solvers. For many years, computer processing unit clock speeds in-

*Part of this section is reprinted with permission from "A Progressive Hedging Approach for Parameter Estimation of Stochastic Nonlinear Programs" by Word, D.P., Watson, J.P., Woodruff, D., and Laird, C.D., 2012. Proceedings of PSE2012, Singapore, Copyright 2012 by Elsevier B.V.

creased dramatically, improving the performance of existing algorithms without any additional effort, but physical hardware limitations are now forcing computer chip manufactures to develop parallel architectures to increase computing power. Hyper-threading and multi-core architectures have become commonplace and can provide significant performance improvements (Schenk et al., 2009), but algorithms must be specifically designed for parallel operation to utilize the hardware's full potential. This algorithm development is essential to continue seeing significant increases in the size and complexity of tractable estimation problems.

Two approaches are commonly used to estimate parameters in dynamic systems. The sequential approach only considers the degrees of freedom as optimization variables. A complete simulation of the problem is performed at each iteration of the optimization algorithm. Calculating the derivative information that is necessary at each time step can be computationally expensive, especially for problems with many degrees of freedom (Hartwich et al., 2011). These calculations are often the dominant computational expense of this approach, and while parallel approaches have been employed to reduce the time required for this step (Diehl et al., 2002; Scheu and Marquardt, 2011; Hartwich and Marquardt, 2010), these techniques can still be inefficient for large-scale problems (Zavala et al., 2008). Alternatively, simultaneous approaches discretize all problem variables and include the discretized problem as constraints in the optimization. This dramatically increases the problem size, but the simulation problem is solved simultaneously with the optimization problem allowing for efficient solution of very large-scale parameter estimation problems (Zavala et al., 2008; Zavala and Biegler, 2006).

Various parallel strategies are being developed to solve nonlinear programs in parallel using the simultaneous approach. Biros and Ghattas (2005) propose a method for optimization of steady-state PDE-constrained problems. This approach uses a reduced space quasi-Netwon method to precondition the full space Karush-Kuhn-Tucker (KKT) system and Krylov iterations to solve the KKT system. This method is fully parallelizable and shows excellent speedup over popular reduced space methods. Zavala et al. (2008) develop a parallel algorithm for large-scale parameter estimation problems. They use an interior-point framework where the dominant computational expense lies in solving the linear KKT system at each iteration of the optimization. They exploit the block bordered diagonal structure inherent in the KKT system of their parameter estimation problems to decompose the problem in parallel using a Schur-complement approach. The Schur-complement is formed and solved in parallel, which dramatically decreases the time required to solve the KKT system. Zhu et al. (2011b) use a similar algorithm for optimal operation under uncertainty. Here, multi-scenario problem formulations are used to capture uncertainty in demand and contractual obligations. The multi-scenario formulation creates a block structure where each scenario is a separate block. A Schur-complement decomposition then exploits this block structure to allow efficient solution in parallel.

In these solution approaches the size of the Schur-complement is related to the number of variables that are common across each block. These variables couple the blocks together, and as the number of these variables grows large, the size of the Schur-complement also becomes large, and the time required to form and solve the Schur-complement can become prohibitive. To counter this, Kang et al. (2013) use a quasi-Newton preconditioned conjugate gradient method that avoids the explicit formation and factorization of the Schur-complement. This allows for significantly better solution times and parallel scalability, especially for problems with significant coupling.

These algorithms show significant promise, but rely upon problems possessing specific structure. If problems do not possess the necessary block structure, these algorithms are not applicable. Additionally, the development of new algorithms is expensive and often outside the expertise of those formulating new problems, so instead of developing new algorithms, it would be preferable to be able to utilize existing solvers. While the use of parallel algorithms is relatively new in the parameter estimation communities, other fields of optimization have been utilizing parallel algorithms for many years (Mulvey et al., 1997).

Stochastic programming is often used for optimization of very large, multi-scenario problems, and many decomposition strategies for solving these problems in parallel have been developed (Mulvey et al., 1997; Rockafellar and Wets, 1991; Ruszczyński, 1993; Petra and Anitescu, 2012). To our knowledge, stochastic programming has not been used for parameter estimation, however, the structure of classical parameter estimation problems is equivalent to that of two-stage stochastic programming problems. This equivalence suggests that stochastic programming techniques can be used for parameter estimation. In this paper, we use the Progressive Hedging algorithm to solve a large-scale dynamic parameter estimation problem.

A typical form of a parameter estimation problem using data from multiple observations can be written as

$$\begin{aligned} \min\ & \sum_{s\in S} P_s f(w_s, \varepsilon_s) \\ \text{s.t.}\ & g(x_s, y_s, w_s, \theta) = 0 \quad \forall\, s \in S \\ & h(y_s, y_s^*, \varepsilon_s) = 0 \qquad \forall\, s \in S. \end{aligned} \tag{6.1}$$

Here, S denotes the set of all observations, P_s, where $\sum_{s\in S} P_s{=}1$, is a weight for each observation, and $f(w_s, \varepsilon_s)$ is a function of goodness-of-fit given the estimated model noise w_s and measurement noise ε_s. The vector of constraints g describes the system model that is a function of unmeasured state variables x_s, model outputs y_s

(measured variables), unknown model noise w_s, and model parameters θ. The model parameters are estimated by fitting the model outputs y_s to known measurements $y_s^\star$. The measurement errors are defined through the vector function h.

A general formulation for the extensive form or deterministic equivalent of a two-stage stochastic programming problem is given as

$$\begin{aligned} \min \quad & \sum_{s \in S} P_s f(v_s, u_s) \\ \text{s.t.} \quad & g(v_s, u_s) = 0 \quad \forall\, s \in S \\ & u_s = \bar{u} \qquad\quad\ \forall\, s \in S. \end{aligned} \tag{6.2}$$

Here, optimal values for first-stage variables $\bar{u}$ are determined while considering uncertainty across a number of possible realizations S. The system model g is a function of first-stage variables u_s and second-stage or recourse variables v_s. While each realization can have its own first-stage variables u_s, they are constrained to be equal at the solution. The objective function usually computes an expected value across all possible realizations or scenarios, where P_s is the probability associated with a particular scenario, and the total probability must equal 1.

The parameter estimation problem in Equation (6.1) is structurally equivalent to the two-stage stochastic programming problem given in Equation (6.2), where the parameters θ are equivalent to first-stage variables $\bar{u}$, and the remaining variables (x_s, y_s, w_s, ε_s) can be considered second-stage variables (v_s). Parameter estimation strives to determine parameter values that are common across all observations, and the measurement and model errors can be considered recourse variables that provide a mechanism for matching the measurements from each individual observation.

The equivalence of these problems is important because a number of stochastic programming decomposition techniques have been developed to handle multi-scenario problems with a very large number of scenarios and decision variables. Progressive Hedging (PH) is one such algorithm that decomposes a stochastic program by scenarios (Rockafellar and Wets, 1991). PH is also particularly amenable to parallelization, and tools exist for automating parallel solution (Watson et al., 2012).

In this section, we demonstrate the potential of PH for the efficient solution of large-scale nonlinear parameter estimation problems. Specifically, we estimate transmission parameters in an infectious disease model using pre-vaccination measles data from 60 cities in England and Wales, made available by Grenfell (2012). Section 6.2 provides a brief overview of SIR infectious disease modeling, develops the disease model formulations solved using NLP algorithms and the PH algorithm, and highlights the equivalence between these formulations. The data used for these estimates is also described here. Section 6.3 outlines the PH algorithm in detail and describes the specific tuning of algorithm parameters performed for this work. Ipopt was used as the PH subproblem solver. Our estimation results are given in Section 6.4 along with some conclusions about these solution approaches and ideas for future work.

6.2 Case Study and Model Formulation

Infectious diseases remain a significant health concern throughout the world, and reliable, mechanistic disease models are desirable to both better understand disease dynamics and plan better response strategies. However, these models require appropriate model parameter values to accurately capture the dynamics observed in reported case data.

Many techniques exist to estimate parameters for temporal dynamics in single cities (Finkenstädt and Grenfell, 2000; Word et al., 2012; Cauchemez and Ferguson, 2008; Hooker et al., 2011), but spatio-temporal dynamics across multiple cities is also important (Xia et al., 2004; Jandarov et al., 2013). In England and Wales prior to vaccination measles was endemic in large cities, but in smaller cities disease fadeout occurred (Xia et al., 2004). Reappearance of the disease would then occur only after a case was imported from a surrounding city where measles was endemic. To capture spatio-temporal dynamics, multi-city models must be developed, but these models can become very large requiring more memory and processing power than a single computer can deliver. To solve these problems efficiently, parallel solution approaches such as PH must be utilized.

Before tackling an estimation problem considering spatial dynamics, we first need a solution approach for solving very large problems. In this work, we demonstrate the use of the progressive hedging algorithm to estimate seasonal transmission parameters for measles using 20 years of reported case counts from 60 cities in England and Wales. While we wish to use PH to estimate parameters in spatially coupled models, here we extend the formulation of Word et al. (2012) to simultaneously estimate transmission parameters for 60 cities without consideration of spatial dynamics.

Word et al. (2012) develop a dynamic, continuous-time SIR compartment-based model that assumes disease progression from the susceptible stage (S) to the infected stage (I) to the recovered and immune stage (R). They then present a simultaneous approach for efficient estimation of seasonally varying transmission parameters. While the formulation presented in Word et al. (2012) was for a single city, we extend this to form a multi-scenario problem where each scenario represents a different city, and all scenarios must have the same parameter values. We make use of the PySP application (Watson et al., 2012), which only requires the explicit formulation of a single-city (or single scenario) model to formulate our problem.

The single-city SIR model used in this work is shown below in Equation (6.3). The corresponding dynamic optimization problem is converted to a large-scale nonlinear programming problem (through the simultaneous discretization approach) using a three-point Radau collocation on finite elements (Zavala, 2008).

$$
\begin{aligned}
\min\ & w_M \int_{t_0}^{t_f} \left(\varepsilon_M(\tau)\right)^2 d\tau + \sum_{i \in T} \left(w_Q(\varepsilon_{Qi}^2)\right) \\
\text{s.t.}\ & \frac{dQ}{dt} = \frac{\beta(y(t))S(t)I(t))}{N} + \varepsilon_M(t) \\
& \frac{dS}{dt} = -\left(\frac{\beta(y(t))S(t)I(t))}{N} + \varepsilon_M(t)\right) + B(t) \\
& \frac{dI}{dt} = \left(\frac{\beta(y(t))S(t)I(t))}{N} + \varepsilon_M(t)\right) - \gamma I(t) \\
& R_i^\star = \mu_i \left(\int_{i-1}^{i} \left(\frac{\beta(y(\tau))S(\tau)I(\tau))}{N} + \varepsilon_M(\tau)\right) d\tau\right) + \varepsilon_{Qi} \quad \forall\, i \in T \\
& \beta_y = \beta^{\text{mag}} \cdot \beta_y^{\text{patt}} \qquad \forall\, y \in \tau \\
& 1.0 = \frac{\sum_{y \in \tau} \beta^{\text{patt}}}{|\tau|} \\
& 0.75 \leq \beta^{\text{mag}} \leq 1.5 \\
& 0 \leq I(t), S(t) \leq N \\
& 0 \leq Q(t).
\end{aligned}
\tag{6.3}
$$

Here, S denotes the number of susceptibles (people with no immunity to the disease), I denotes the number of infectives (people who possess the disease and are infectious), N denotes the total population, and $\beta(t)$ denotes the time-varying transmission parameter. The function $y(t)$ maps the overall time horizon into the elapsed time within the year, making $\beta(y(t))$ a seasonal transmission parameter with periodicity of one year. The parameter B denotes the number of reported births, and the recovery rate ($\gamma = 1/14$) is given as a known scalar input. The variable ε_M represents the dynamic model noise, which is assumed to be normally distributed. The index i denotes a point in time within the set of data reporting intervals T, μ denotes a reporting factor accounting for under-reporting over the time interval spanning $i-1$ to i, and $R_i^\star$ denotes the actual reported incidence over a given time interval. The

ε_{Qi} term represents the measurement noise, and w_M and w_Q represent weights for the model and measurement noise terms, respectively. These weights are set to be proportional to the inverse of the assumed variance of the error terms. The scalar β^{mag} specifies the magnitude of the transmission parameter and is constrained to be within values considered reasonable R_0 values for measles (Anderson and May, 1991). The yearly pattern of β is specified by β^{patt} and must have a mean of 1.0. The yearly set of discretizations τ has the cardinality $|\tau|$.

The data used in this work is from 60 cities in England and Wales and contains biweekly reported measles case counts by city for the years 1944 through 1963. The number of births are reported per year by city, and we assume the births to be uniform throughout the year. The population is assumed to be constant for all cities. Since not every measles infection is reported, case counts are under-reported. We use a straightforward approach that is described elsewhere (Word et al., 2012) to estimate a linearly varying reporting fraction for London. This same reporting fraction is then used for all cities. Ideally, a reporting fraction would be estimated for each city, but this is infeasible due to the low number of cases reported in many smaller cities.

The differential equations from (6.3) are discretized using a three-point Radau collocation on finite elements as described in Section 3.3. Two discretizations were used for our work. The low-discretization case used the same number of finite elements as there were reporting intervals in the data, which for this data results in 26 finite elements per year. The high-discretization case used 4 times the number of finite elements as there were reporting intervals, or 104 finite element per year.

6.3 Estimation Approach

The single-city problem shown in Equation (6.3) was formulated in the algebraic modeling language Pyomo (Hart et al., 2011, 2012). This model is for a single city and a single data set. We then specify a scenario tree in PySP (Python-based Stochastic Programming) (Watson et al., 2012), a python based modeling and solver library for stochastic programming, and PySP automatically converts the single-city problem into the multi-city problem. PySP and Pyomo are both part of the open-source Coopr software package released by Sandia National Laboratories. More details of this software can be found in Section 3.1. PySP uses a single scenario model that is formulated in Pyomo, a scenario tree, and multiple data files to construct a multi-scenario problem. The problem can then be solved using two approaches.

The first and preferred approach considers the extensive form of the multi-scenario problem. The extensive form of the multi-scenario extension of (6.3) is

$$
\begin{aligned}
\min \quad & \sum_{c \in C} P_c \left(w_M \int_{t_0}^{t_f} \left(\varepsilon_M(\tau)\right)^2 d\tau + \sum_{i \in T} \left(w_Q(\varepsilon_{Qi}^2) \right) \right) & \\
\text{s.t.} \quad & \frac{dQ^c}{dt} = \frac{\beta^c(y(t)) S^c(t) I^c(t))}{N^c} + \varepsilon_M^c(t) & \forall\, c \in C \\
& \frac{dS^c}{dt} = -\left(\frac{\beta^c(y(t)) S^c(t) I^c(t))}{N^c} + \varepsilon_M^c(t) \right) + B^c(t) & \forall\, c \in C \\
& \frac{dI^c}{dt} = \left(\frac{\beta^c(y(t)) S^c(t) I^c(t))}{N^c} + \varepsilon_M^c(t) \right) - \gamma I^c(t) & \forall\, c \in C \\
& R_i^{c\star} = \mu_i \left(\int_{i-1}^{i} \left(\frac{\beta^c(y(\tau)) S^c(\tau) I^c(\tau))}{N^c} + \varepsilon_M^c(\tau) \right) d\tau \right) + \varepsilon_{Qi}^c & \forall\, i \in T,\ \forall\, c \in C \\
& \beta_y^c = \beta^{c\text{mag}} \cdot \beta_y^{c\text{patt}} \qquad \forall\, y \in \tau & \forall\, c \in C \\
& 1.0 = \frac{\sum_{y \in \tau} \beta^{c\text{patt}}}{|\tau|} & \forall\, c \in C \\
& 0.75 \le \beta^{c\text{mag}} \le 1.5 & \forall\, c \in C \\
& 0 \le I^c(t), S^c(t) \le N^c & \forall\, c \in C \\
& 0 \le Q^c(t) & \forall\, c \in C \\
& \beta = \beta^c & \forall\, c \in C.
\end{aligned}
\tag{6.4}
$$

Here, all variables are defined as in (6.3) except that variables are now defined for each city c from the set of all cities C. The objective term P_c allows for individual cities to be weighted differently, but in our work all P_c's are equal. The last constraint forces the first-stage variables β^c to all converge to the same solution β. In this work we use the interior-point algorithm IPOPT to solve this problem. For more details about this algorithm please see Section 3.2 and the literature (Wächter and Biegler, 2006).

While the extensive form of the problem should be solved if possible, for many scenarios and very large problems, this approach can not only require an excessive amount of time to converge but also exceed the memory capacity of a single computer. These limitations require the use of alternative approaches. In addition to formulating the extensive form of problems, PySP also allows for solution utilizing the Progressive Hedging (PH) algorithm of Rockafellar and Wets (Rockafellar and Wets, 1991). Here, we use IPOPT as the PH subproblem solver.

The PH algorithm is outlined below. Weights P_c are placed on each observation, C is the set of all observations, w_c^k is an algorithm parameter that is updated at each iteration k, and ρ is an algorithm tuning parameter. The weighted average of the estimated parameters from each observation is $\bar{\beta}$. Using the single-city problem formulation (6.3):

1. Initialize the iterate $k \leftarrow 0$ and let $w_c^k = 0$

2. For all scenarios $c \in C$, solve (6.3) for the model parameters β_c:

$$\beta_c^0 \leftarrow \text{argmin} f_c(\varepsilon)$$

3. Update the iterate, $k \leftarrow k + 1$, and update $\bar{\beta}$:

$$\bar{\beta}^{k-1} \leftarrow \textstyle\sum_{c \in C} P_c \beta_c^{k-1} \text{ where } \sum_{c \in C} P_c = 1$$

4. For all scenarios $c \in C$, update w:

$$w_c^k \leftarrow w_c^{k-1} + \rho_c(\beta_c^{k-1} - \bar{\beta}_c^{k-1})$$

5. Modify the original problem by augmenting the objective function $f_c(\varepsilon)$ from (6.3) to:

$$f_c(\varepsilon) + w_c^k \beta_c + \tfrac{\rho_c}{2} \|\beta_c - \bar{\beta}^{k-1}\|^2$$

6. For all scenarios $c \in C$, solve the modified problem for the model parameters β_c:

$$\beta_c^k \leftarrow \text{argmin } f_c(\varepsilon) + w_c^k \beta_c + \tfrac{\rho_c}{2} \|\beta_c - \bar{\beta}^{k-1}\|^2$$

7. If termination criterion not met, go back to step 3.

Steps 2 and 6 are computationally expensive, but these steps can be performed in parallel, which allows for significant performance improvements.

Strategies exist for setting the values for ρ_c, and choosing reasonable values for these parameters can significantly effect run-times (Watson and Woodruff, 2011). Watson and Woodruff (2011) suggest ρ should be proportional to, or at least correlated with, the rate of change in the objective function with changes in the first-stage variables. For objective variables with no interaction (second-stage variables), this is given by the cost coefficient (the objective function given in Equation (6.3)). PH is trying to converge the model parameters β_c from each observation to the same solution.

In this work, ρ_c is set to be the same for every observation. To balance the objective function contributions from first-stage and second-stage variables, we choose a value that is on the same order of magnitude as the cost coefficient. Setting these parameter values too high can lead to convergence to suboptimal solutions, while setting these values too low can lead to excessively long convergence times.

The subproblem solver Ipopt also has algorithm parameters that can be set to decrease run times. When a good initialization is available, setting the initial value of the barrier parameter (μ in Equation 3.3) to be lower than the default value can reduce the number of iterations necessary to converge the problem and thus reduce run times. However, if a good initialization is not available, lowering this initial value can significantly increase run times. In this work, the default initial value is used for the zeroth iteration of PH. For the second PH iteration, μ_0 is set to $1{\times}10^{-3}$, and for all subsequent PH iterations $\mu_0 = 1{\times}10^{-6}$.

The seasonal transmission parameter estimated using both the extensive form and PH approaches are shown in Figures 6.1 and 6.2. The problem sizes and solution times for both approaches are shown in Tables 6.1 and 6.2.

6.4 Results and Conclusions

The estimated values of the seasonal transmission parameters for the low and high discretization problems are shown in Figures 6.1 and 6.2. The extensive form and the PH approach yield nearly identical solutions. This solution is also consistent with published values Finkenstädt and Grenfell (2000). PH was terminated once the sum of differences between the first stage variables and their average reached a value less than 1×10^{-4}, indicating logical convergence.

All timing results were obtained using the Red Mesa supercomputing cluster at Sandia National Lab. This cluster is made up of computing nodes, each with two, 2.93 GHz quad-core, Nehalem X5570-processors, giving 8 computing cores per node. Each node has 12 GB of DDR3 RAM. Results for the low discretization problem given in Table 6.1 show that the extensive form solution requires a total of 4.2GB of RAM. We wish to solve problems with hundreds of observations, and problems of this size will not only require very long solution times but also more memory than would be available on a single computing node. Using Progressive Hedging in serial requires more time than solving the extensive form of the problem, but less memory is required. The parallel solution of this problem using PH shows acceptable speed-up, and memory requirements remain low for each compute node, making it possible to tackle much larger problems.

Table 6.2 shows statistics for the high-discretization problem. The extensive form of the problem could not be solved due to the problem size exceeding the memory capacity of the computing node, and the solution time using progressive hedging in

	Extensive Form	PH (serial)	PH (60 processors)
Jacobian Nonzeros	2600702	42248	42248
Variables	627568	10457	10457
Constraints	596341	9910	9910
Solver Time (min)	5.2	14.8	-
PySP Time (min)	3.8	24.2	-
Total Time (min)	9.0	39	1.9
Solver RAM (GB)	1.9	<0.05	<0.05
PySP RAM (GB)	2.3	1.8	<0.6
Total RAM (GB)	4.2	1.9	<0.7

Table 6.1: Timing and Memory Results For the Low-Discretization Case

serial exceeded our 2 hour time limit. However, using progressive hedging in parallel allowed us to find a solution in less than 8 minutes while never exceeding 2GB of memory usage on any single computing node.

	Extensive Form	PH (serial)	PH (60 processors)
Jacobian Nonzeros	9995102	166529	166529
Variables	2405968	40098	40098
Constraints	2281141	37992	37992
Total Time (min)	-	-	7.5
Total RAM (GB)	>12	>7	<2

Table 6.2: Timing and Memory Results For the High-Discretization Case

Eqs. 6.1 and 6.2 demonstrate that two-stage stochastic programming problems are structurally equivalent to nonlinear programming parameter estimation problems. This similarity allows large-scale nonlinear parameter estimation problems to be solved using stochastic programming decomposition techniques. Our results demonstrate that Progressive Hedging can be used to solve very large parameter estimation problems in parallel with significant reductions in solution times and memory requirements.

Further improvements will allow even greater speed-up for parallel PH. Currently, if even one scenario requires significantly longer to converge than the rest of the scenarios, a considerable number of computing nodes remain idle for extended periods of time. Asynchronous approaches being studied will reduce the number of idle processors by allowing the majority of computing nodes to continue with the next iteration of the algorithm using an altered update formula. Improved strategies for the selection of the algorithm parameter ρ could also further reduce solution times.

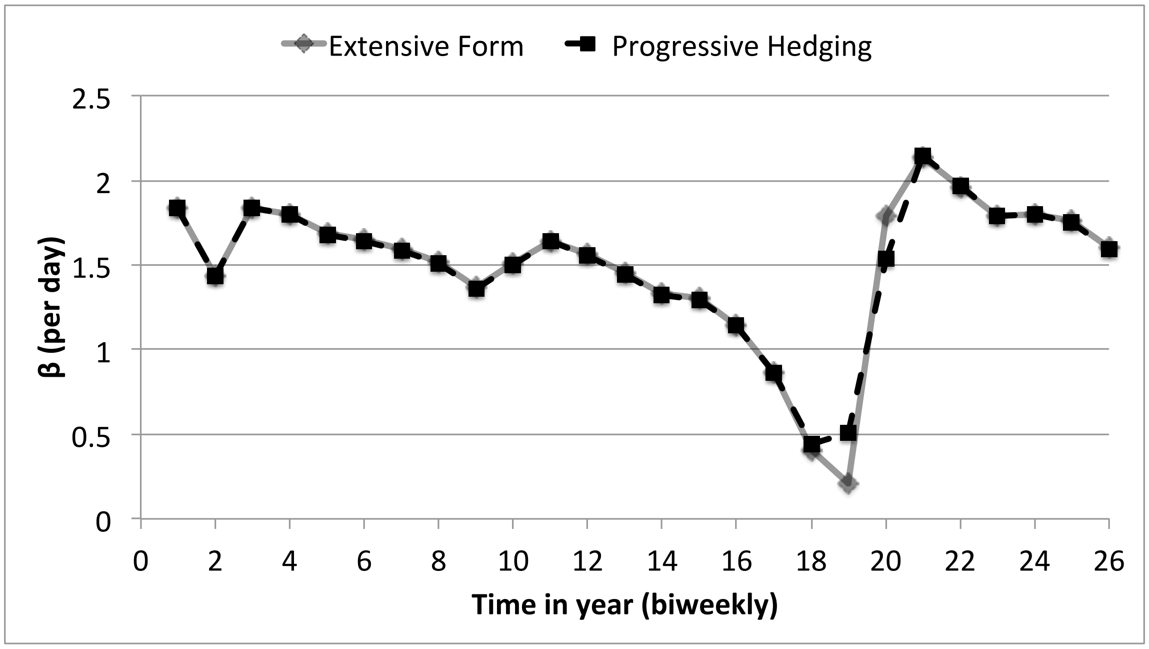

Figure 6.1: The seasonal transmission profile β estimated for the low-discretization problem using the extensive form approach (diamonds) and progressive hedging approach (squares).

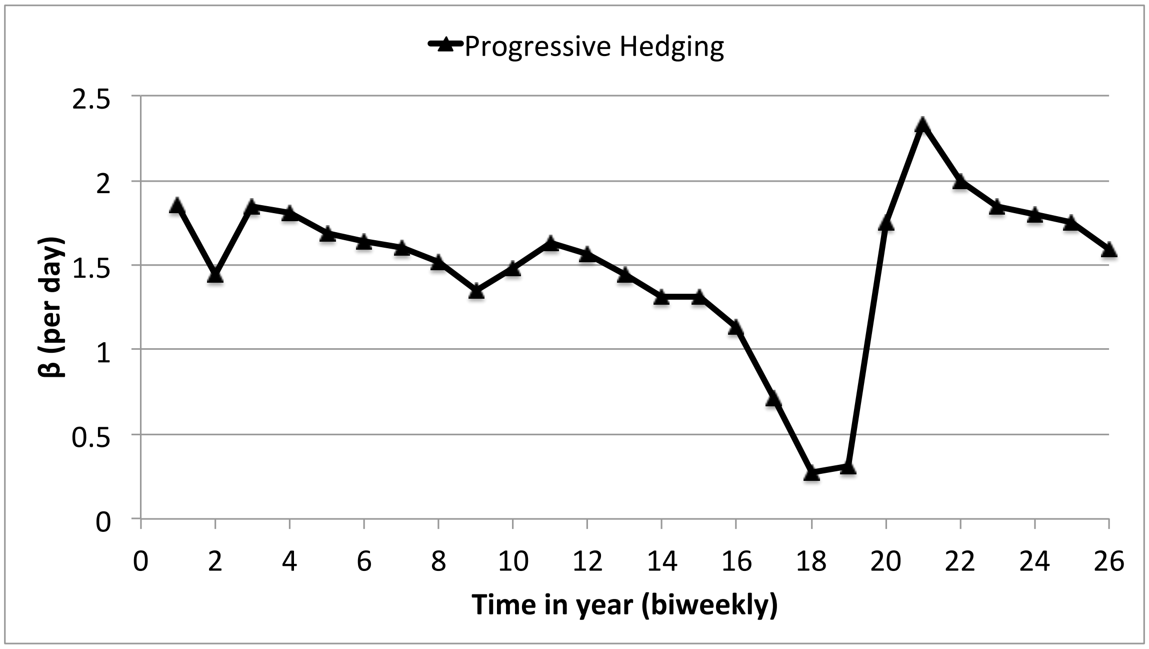

Figure 6.2: The seasonal transmission profile β estimated for the high-discretization problem using the progressive hedging approach.

7. PARALLEL DYNAMIC OPTIMIZATION [*]

In Section 6 we presented a scenario-based decomposition approach for infectious disease models. This strategy essentially separated a large, multi-city estimation problem into subproblems of fewer (or even individual) cities. Another strategy is to decompose the problem in time. Our estimation problems are dynamic, and the structure of the problem in time is conducive to certain decompositions.

This section presents a decomposition strategy applicable to DAE constrained optimization problems. A common solution method for such problems is to apply a direct transcription method and solve the resulting nonlinear program using an interior-point algorithm. For this approach, the time to solve the linearized KKT system at each iteration typically dominates the total solution time. In our proposed method, we exploit the structure of the KKT system resulting from a direct collocation scheme for approximating the DAE constraints in order to compute the necessary linear algebra operations on multiple processors. This approach is applied to find the optimal control profile of a combined cycle power plant with promising results on both distributed memory and shared memory computing architectures with speedups of over 50 times possible.

7.1 Parallel Optimization of Dynamic Systems

Optimization of dynamic systems has proven to be an effective method for improving operation and profits in the chemical process industry (Scheu and Marquardt, 2011; Diehl et al., 2002; Zavala et al., 2008; Zhu et al., 2010; Tanaka and Martins, 2011). The success of these methods has led to the continued growth of these sys-

[*]Part of this section is reprinted with permission from “Efficient Parallel Solution of Large-Scale Nonlinear Dynamic Optimization Problems” by Word, D.P., Kang, J., Akesson, J., and Laird, C.D., 2013. Submited to Computation Optimization and Applications.

tems to improve model rigor and increase the scope of the optimization problem, however the solution of very large-scale models remains challenging (Hartwich and Marquardt, 2010). Concurrently, the advances in computing clock rates that we once took for granted have slowed dramatically. Computer chip design companies have instead focused on development of parallel computing architectures (Zhu et al., 2009). These new architectures require advanced algorithms to exploit their parallelism. Furthermore, the successful use of advanced solution approaches within industrial settings requires that these algorithms be interfaced with effective problem formulation tools. Modern object-oriented modeling languages allow for rapid creation of complex dynamic optimization problems and reduce the burden of model development, optimization problem formulation, and solver interfacing. These also ease the construction of complicated optimization problems, while making it easier to construct intractably large problems for serial algorithms. There is a need for the development of advanced parallel algorithms for dynamic optimization that can interface with modern modeling languages and utilize parallel computing architectures to efficiently solve these large-scale problems. This section presents a decomposition approach for efficient, parallel solution of nonlinear dynamic optimization problems formulated using the Modelica-based JModelica.org platform (Åkesson et al., 2010) and the optimization package CasADi (Andersson et al., 2012).

There are three common approaches for solution of dynamic optimization problems. With sequential techniques (also called control vector parameterization), the control variables are discretized, and the optimizer only sees these discretized degrees of freedom. An integrator then converges the model at each iteration with calculation of derivative information performed through various techniques including integrating sensitivity or adjoint equations. The integration of the model and calculation of the derivative information is often the dominant computational expense of this technique

(Hartwich et al., 2011), and multiple approaches are being explored to improve solution times through parallelization of these computations (Scheu and Marquardt, 2011; Hartwich and Marquardt, 2010; Diehl et al., 2002; Leineweber et al., 2003). Multiple shooting techniques combine aspects of both sequential and simultaneous approaches. These techniques discretize time into multiple stages and the control variables are parameterized using a finite set of control parameters in each stage. The system is then solved on each stage, where each stage is an initial value problem. Equality constraints are added to enforce continuity between the stages (Biegler and Grossmann, 2004).

In contrast to sequential approaches, simultaneous approaches discretize all problem variables, not just control variables, and include the discretized model as constraints in a large-scale optimization problem (Biegler and Grossmann, 2004). Collocation-based methods discretize the entire problem to form a large-scale NLP, where the profiles are approximated using polynomials on finite elements. While the size of the optimization problem is significantly larger in this case, there is potential for improved performance since the simulation problem (represented by the discretized equality constraints) is solved simultaneously with the optimization problem. Even though such problems are very large, they are sparse and inherently structured as a result of the discretization. In this section, we consider efficient solution of simultaneous collocation-based discretization approaches.

One hurdle in using simultaneous approaches in the past has been the burden on the modeler associated with the manual discretization of the model – a process that is typically tedious and error-prone. However, packages are available in modern modeling languages such as the Modelica-based JModelica.org platform (Association et al., 2007; Åkesson et al., 2010), Pyomo (Hart et al., 2012), ACADO (Houska et al., 2011), and DynoPC (Lang and Biegler, 2007) that allow straightforward declaration of dynamic equations and provide automatic discretization of these equations using direct collocation methods. This significantly reduces the burden on the modeler when using simultaneous solution approaches.

Nonlinear interior-point methods provide an excellent framework for specialized solution of these discretized dynamic optimization problems. These methods have proven to be effective tools to solve many very large-scale optimization problems (Zavala and Biegler, 2006; Zhu et al., 2010; Laird et al., 2005; Word et al., 2012). The dominant computational expense in these methods usually lie in the solution of the augmented system, or KKT system, a linear system that results from the application of Newton's method to the primal-dual form of the KKT conditions of the barrier subproblem. The KKT system retains consistent structure from iteration to iteration, and various approaches can be utilized to reduce the time required to solve this system.

Amestoy et al. (2000) developed a parallel distributed memory multifrontal approach for the solution of sparse linear equations that includes an asynchronous parallel algorithm for efficient numerical pivoting. This algorithm showed speedup of more than 7 on test problems, but algorithm performance was not evaluated using a large number of processors because suitable test problems were unavailable.

Scott (2003) presented parallel general-purpose multi-frontal codes to solve large sparse systems of linear equations. These codes include the serial solution of a so-called interface problem, and the size and subsequent time for solution of this problem heavily influences the overall performance. This serial component limits the scalability of the approach to a relatively small number of processors, but the results still show a 5 times speedup on 8 processors compared with serial direct solvers.

Schenk and Gärtner (2004) address issues to improve scalability and robustness of sparse direct factorization on shared memory multiprocessor architectures. To balance trade-off between performance and robustness, they employ block supernode diagonal pivoting. While this approach is not guaranteed to always be efficient, timing results are promising for some problems. Scale-up of this approach is limited by the fact that it is designed for shared memory architectures.

Kocak and Akay (2001) explore a decomposition approach for solution of general linear systems using a Schur-complement. They investigate the use of an analysis step to determine problem structure that can be exploited using this decomposition and highlight that when exploiting problem structure for the Schur-complement decomposition, the amount of coupling between blocks makes a significant impact on algorithm performance. For problems lacking clear structure, determining how to effectively decompose the problem is vital to maintain algorithm efficiency but can be computationally expensive. On the other hand, certain problem classes contain well-defined coupling by definition and offer an intuitive path for block structure decompositions.

An alternative to using parallel direct solvers that handle general problem structures is to focus on problem classes with a particular structure and develop algorithms that specifically exploit that structure. By requiring specific structure in problem formulations, the structure is predetermined so the cost of structural analysis can be avoided. This knowledge can then be exploited for problem level decomposition and parallel solution of these decomposed systems. Schur-complement methods have been used for many years to decompose and solve large-scale, structured, linear systems, and efficient parallel algorithms have been developed (Kocak and Akay, 2001).

DeMiguel and Nogales (2008) have established a theoretical relationship between bilevel decomposition algorithms and Schur-complement interior-point methods. This helps bridge the gap between the local convergence theory of bilevel decomposition algorithms and Schur interior-point methods. Additionally, this work shows how Schur-complement interior-point methods can be modified to allow solution of problems where the Schur-complement is not generally invertible (DeMiguel and Nogales, 2008). Goulart et al. (2008) have applied robust optimization techniques to reparameterize linear discrete-time optimal state feedback problems as convex programs. A primal-dual interior-point solver is then used to solve this convex program. They highlight that it is straightforward to parallelize their algorithm if a Schur-complement decomposition is used to solve the linear KKT system at each iteration of the interior-point algorithm. This solution approach is shown to be efficient even though they do not implement their algorithm in parallel. Zavala et al. (2008) adapted the interior-point solver IPOPT to use a Schur-complement decomposition approach to solve the linear KKT system in parallel by exploiting block structure in multi-scenario parameter estimation problems. This approach has proven effective for solving several large-scale case studies (Laird and Biegler, 2008; Zavala et al., 2008). Zhu and Laird (2008) present a similar algorithm for optimal control under uncertainty. In Zhu et al. (2011b) this algorithm is used to determine optimal control of a cryogenic air separation where demand and contractual obligation uncertainties are captured using a multi-scenario formulation. For these multi-scenario problems, individual scenarios can be decomposed into separate blocks where only variables that are common across multiple scenarios prevent the blocks from being fully independent. These coupling variables are permuted into the Schur-complement and dictate its size. The Schur-complement is solved in serial, and as the number of coupling variables and the size of the Schur-complement grows, this computational cost can become significant. However, for problems with a low number of coupling variables this approach shows significant performance improvements over full-space solution approaches (Zhu et al., 2011b).

In this section we make use of the Modelica-based open source software JModelica.org (Åkesson et al., 2010) and CasADi (Andersson et al., 2012), to transform high-level descriptions of nonlinear dynamic optimization problems into algebraic nonlinear programming problems through a direct collocation approach. This creates a block-banded structure in the KKT system where each finite element forms a block, and the system can be decoupled at finite element boundaries. The literature details a number of strategies exploiting this finite element block structure in serial (Cervantes and Biegler, 2000; Cervantes et al., 2000; Biegler et al., 2002; Rao et al., 1998). When applying a nonlinear interior-point method to solve the optimization problem, the dominant computational expense is the solution of the KKT system that must be solved at each iteration to produce the steps in the primal and dual variables. The block-banded structure is decomposed by forming a Schur-complement with respect to the state continuity equations. The size of the Schur-complement depends on the number of state variables and the number of processors used in the decomposition, making this approach most favorable for problems with significantly fewer state variables than algebraic variables.

Building on results in Laird et al. (2011), we develop an interior-point algorithm for the solution of large-scale nonlinear dynamic optimization problems. We use a Schur-complement decomposition approach that exploits the block-structure inherent in the discretized optimization problem to allow solution of the problem in parallel. The performance of our decomposition algorithm is demonstrated on an optimal control problem for the start-up of a combined cycle power plant, where we achieve a speedup of over 50 times.

In Section 7.2 we describe the interior-point algorithm used to solve our optimization problems, and in Section 7.3 we describe the discretization strategy employed to transform dynamic problems into nonlinear programming problems. Section 7.4 shows how we decouple individual finite element blocks and use a Schur-complement decomposition to solve for the primal and dual variable steps in the interior-point algorithm. We describe our software implementation in Section 7.5 and outline the formulation of test problems in Section 7.6. We display the performance of our algorithm on these problems in Section 7.7. Conclusions and future work are summarized in Section 7.8.

7.2 Interior-Point Algorithm

The following discussion of the interior-point algorithm used in this section repeats much of the discussion presented in Section 3. It is repeated here do to its immediate relevance in understanding the decomposition algorithm presented here. This algorithm considers nonlinear problems of the form

$$\min \; f(x) \tag{7.1}$$

$$\text{s.t. } c(x) = 0 \tag{7.2}$$

$$\underline{x} \leq \; x \leq \overline{x}, \tag{7.3}$$

with n variables and m equality constraints, where $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $c : \mathbb{R}^n \rightarrow \mathbb{R}^m$ are assumed to have continuous first and second derivatives. The vectors $\underline{x}$ and $\overline{x}$ are the set of lower and upper variable bounds for x respectively. This problem is solved using an interior-point method with a filter-based line-search based on that described in Wächter and Biegler (2006). A detailed description of the algorithm and its derivation can be found there, and here we only describe the steps necessary to explain our parallel decomposition approach. As an interior-point method, variable bounds are removed from the constraints by adding a log penalty to the objective and forming the barrier subproblem,

$$\begin{aligned} \min \quad & f(x) - \mu \sum_{i=1}^{n} \ln(\overline{x}^{(i)} - x^{(i)}) - \mu \sum_{i=1}^{n} \ln(x^{(i)} - \underline{x}^{(i)}) \\ \text{s.t.} \quad & c(x) = 0, \end{aligned} \tag{7.4}$$

where μ is the barrier parameter for a single barrier iteration, and (i) denotes the i^{th} element of the vectors of length n.

The Lagrangian of the barrier subproblem (7.4) can then be written as,

$$\mathcal{L} = f(x) - \mu \sum_{i=1}^{n} \ln(\overline{x}^{(i)} - x^{(i)}) - \mu \sum_{j=1}^{n} \ln(x^{(j)} - \underline{x}^{(j)}) + \lambda^T c(x), \tag{7.5}$$

where λ is the vector of equality constraint multipliers. The first order optimality conditions are then,

$$\begin{aligned} \nabla_x \mathcal{L} = \nabla_x f(x) + \mu(\overline{X})^{-1} e - \mu(\underline{X})^{-1} e + \nabla_x c(x) \lambda = 0 \\ c(x) = 0, \end{aligned} \tag{7.6}$$

with $\overline{X}$=diag($\overline{x}-x$) and $\underline{X}$=diag($x-\underline{x}$). We form the primal-dual formulation by introducing the variables $\overline{d}=\mu[\overline{X}]^{-1}e$ and $\underline{d}=\mu[\underline{X}]^{-1}e$. The algorithm enforces $(\overline{x}-x)\geq 0$ and $(x-\underline{x})\geq 0$, which makes the new variables $\overline{d}, \underline{d} \geq 0$. Including these new variables gives the following system of equations:

$$\begin{aligned} \nabla_x \mathcal{L} = \nabla_x f(x) + \overline{d} - \underline{d} + \nabla_x c(x) \lambda = 0 \\ c(x) = 0 \\ \underline{X}\,\underline{d} - \mu e = 0 \\ \overline{X}\,\overline{d} - \mu e = 0. \end{aligned} \tag{7.7}$$

These equations are solved for a particular value of μ using a modified Newton's method. For each iteration k of the Newton's method, the following linear system must be solved:

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}(x^k) & \nabla_x c(x^k) & -I & I \\ [\nabla_x c(x^k)]^T & 0 & 0 & 0 \\ \underline{D}^k & 0 & \underline{X}^k & 0 \\ -\overline{D}^k & 0 & 0 & \overline{X}^k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta \lambda^k \\ \Delta \underline{d}^k \\ \Delta \overline{d}^k \end{bmatrix} = - \begin{bmatrix} \nabla_x f^k + \overline{d}^k - \underline{d}^k + \nabla_x c(x^k)\lambda \\ c(x^k) \\ \underline{X}^k \underline{d}^k - \mu e \\ \overline{X}^k \overline{d}^k - \mu e \end{bmatrix}. \tag{7.8}$$

Here, Δx^k, $\Delta\lambda^k$, $\Delta\underline{d}^k$, and $\Delta\overline{d}^k$ are the full steps for each of the respective variables, $\underline{D}^k$=diag($\underline{d}^k$), and $\overline{D}^k$=diag($\overline{d}^k$).

The augmented form, a symmetric system, is obtained by multiplying the third block row by $(\underline{X}^k)^{-1}$, the fourth block row by $(-\overline{X}^k)^{-1}$, and adding these rows to the first block row. This gives

$$\begin{bmatrix} H^k & \nabla_x c(x^k) \\ [\nabla_x c(x^k)]^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta \lambda^k \end{bmatrix} = - \begin{bmatrix} \tilde{r}^k_x \\ c(x^k) \end{bmatrix}, \tag{7.9}$$

where,

$$H^k = \nabla^2_{xx}\mathcal{L}(x^k) + (\underline{X}^k)^{-1}\underline{D}^k + (\overline{X}^k)^{-1}\overline{D}^k \tag{7.10}$$

and

$$\tilde{r}_x^k = \nabla_x f^k + \nabla_x c(x^k)\lambda - (\underline{X}^k)^{-1}\mu e + (\overline{X}^k)^{-1}\mu e. \tag{7.11}$$

This interior-point algorithm employs a filter-based line-search strategy that requires the generated step to be a descent direction. This is ensured if the following inertia condition is satisfied (Forsgren et al., 2002),

$$\text{Inertia}(K) = (n, m, 0). \tag{7.12}$$

The inertia is the number of positive, negative, and zero eigenvalues of the matrix K, n is the number of variables, m is the number of equality constraints, and

$$K = \begin{bmatrix} H^k & \nabla_x c(x^k) \\ [\nabla_x c(x^k)]^T & 0 \end{bmatrix}. \tag{7.13}$$

We wish to use this algorithm to solve general non-convex NLPs. To ensure descent directions for these problems, we may need to modify the linear system (7.13) utilizing inertia correction. The modified linear system is

$$\begin{bmatrix} H^k + \delta_H I & \nabla_x c(x^k) \\ [\nabla_x c(x^k)]^T & -\delta_c I \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta \lambda^k \end{bmatrix} = - \begin{bmatrix} \tilde{r}_x^k \\ c(x^k) \end{bmatrix}. \tag{7.14}$$

Here, δ_H and δ_c are set to zero except when their values must be increased to satisfy the inertia condition. This system is solved for each interior-point iteration to calculate the full steps in x and λ. The necessary algebra is then performed to calculate the steps in $\Delta\overline{d}$ and $\Delta\underline{d}$, and a line-search is used to ensure that the steps taken in each of these variables are suitable.

In this algorithm, the most computationally expensive steps are solving the linear system (7.14) and calculating the residuals ($|c(x^k)|$), gradients ($\nabla_x f^k$ and $\nabla_x c(x^k)$), and Hessian H^k. For the dynamic problems addressed in this section, inherent structure exists that allows for decomposition and efficient parallel solution of this linear system and parallel evaluation of these functions. In the next section, we describe the transcription approach we use for dynamic problems that induces the structure of these problems.

7.3 Model Transcription

The dynamic optimization problems considered in this section are based on differential algebraic equation (DAE) models of the form

$$\min_u \int_{t_0}^{t_f} L(x,u,y)\,dt \tag{7.15}$$

s.t.

$$F(\dot{x},x,u,y) = 0, \tag{7.16}$$

$$x(t_0) = x_0 \tag{7.17}$$

$$\underline{z} \le z \le \overline{z} \tag{7.18}$$

where $\dot{x} \in R^{n_x}$ are the state derivative variables, $x \in R^{n_x}$ are the state variables, $u \in R^{n_u}$ are the control input variables, and $y \in R^{n_y}$ are the algebraic variables. The vector z contains all problem variables ($\dot{x}$, x, u, and y), and $\underline{z}$ and $\overline{z}$ are the lower and upper bounds respectively on z. It is assumed that the DAE is of index 1, and index reduction techniques can be used to meet this assumption (Mattsson and

Söderlind, 1993).

The optimization problem is discretized using a simultaneous transcription method based on finite elements, with Radau collocation points. See, e.g. (Biegler, 2010) for a recent monograph. Lagrange polynomials are used to approximate the state, algebraic and control input profiles.

The optimization mesh is defined by n_e finite elements, with normalized lengths h_i, i=$1, ..., n_e$, where $\sum_{i=1}^{n_e} h_i = 1$. The starting point of each finite element is then given by

$$t_i = t_0 + (t_f - t_0) \sum_{k=1}^{i-1} h_k \ \ \forall\, i = 1, ..., n_e, \tag{7.19}$$

and the collocation points are given by

$$t_{i,j} = t_0 + (t_f - t_0) \left(\sum_{k=1}^{i-1} h_k + \tau_j h_i \right) \ \ \forall\, i = 1, ..., n_e, \ \ j = 1, ..., n_c, \tag{7.20}$$

where $\tau_j \in (0, 1]$, j=$1, ..., n_c$ are the Radau collocation points.

At each collocation point, the discretized variable vectors, $\dot{x}_{i,j}$, $x_{i,j}$, $u_{i,j}$, and $y_{i,j}$ for i=$1, ..., n_e$ and j=$1, ..., n_c$, are introduced. In addition, state variables at the beginning of each finite element, $x_{i,0}$ for i=$1, ..., n_e$, are introduced. In each finite element, the differentiated variables are approximated by

$$x(t) = \sum_{k=0}^{n_c} x_{ik} L_k^{n_c+1} \left(\frac{t - t_i}{(t_f - t_0) h_i} \right), \ t \in [t_i, t_{i+1}] \tag{7.21}$$

where $L_j^{n_c+1}(\tau)$ are Lagrange polynomials of order n_c which are computed based on the points $\tau_0, ..., \tau_{n_c}$, with τ_0=0. Accordingly, the expressions for the derivatives $\dot{x}_{i,j}$

are given by

$$\dot{x}_{i,j} = \frac{1}{(t_f - t_0)h_i} \sum_{k=0}^{n_c} x_{i,k} \dot{L}_k^{n_c+1}(\tau_j),\ i = 1, ..., n_e,\ j = 1, ..., n_c. \tag{7.22}$$

At each collocation point, $t_{i,j}$, the DAE relation (7.16)

$$F(\dot{x}_{i,j}, x_{i,j}, u_{i,j}, y_{i,j}) = 0,\ i = 1, ..., n_e,\ j = 1, ..., n_c \tag{7.23}$$

holds. In addition, continuity constraints for the state variable profiles are enforced

$$x_{i-1,n_c} = x_{i,0},\ i = 2, ..., n_e \tag{7.24}$$

as well as the initial conditions $x_{1,0}{=}x_0$. Notice that the relation (7.24) holds since for Radau collocation points, $\tau_{n_c}{=}1$.

We denote the equations (7.22) and (7.23) in residual form for each finite element i by $R_i{=}R(z_i){=}0$, where $z_i^T{=}[x_{i,0}^T \dot{x}_{i,1}^T, x_{i,1}^T, u_{i,1}, y_{i,1}^T, \ldots, \dot{x}_{i,n_c}^T, x_{i,n_c}^T, u_{i,n_c}, y_{i,n_c}^T]$.

The cost function (7.15) is discretized using a quadrature formula

$$\sum_{i=1}^{n_e} \sum_{j=1}^{n_c} w_j L(x_{i,j}, u_{i,j}, y_{i,j}) = f(z) \tag{7.25}$$

where w_j are the Radau quadrature weights.

The discretized optimal control problem can now be written in the form

$$\min_{z}\ f(z) \tag{7.26a}$$

$$\text{s.t.}\ c(z) = 0 \tag{7.26b}$$

$$\underline{z} \leq z \leq \overline{z} \tag{7.26c}$$

where

$$z = \begin{bmatrix} z_1 \\ \vdots \\ z_{n_e} \end{bmatrix}, \qquad z_i = \begin{bmatrix} x_{i,0} \\ \dot{x}_{i,1} \\ x_{i,1} \\ u_{i,1} \\ y_{i,1} \\ \vdots \\ \dot{x}_{i,n_c} \\ x_{i,n_c} \\ u_{i,n_c} \\ y_{i,n_c} \end{bmatrix} \forall\, i = 1, ..., n_e, \quad \text{and} \quad c(z) = \begin{bmatrix} \overline{G} z_1 - x_0 \\ R(z_1) \\ \underline{G} z_1 + \overline{G} z_2 \\ \vdots \\ \underline{G} z_{n_e-1} + \overline{G} z_{n_e} \\ R(z_{n_e}) \end{bmatrix}. \tag{7.27}$$

Here, $\underline{z}$ and $\overline{z}$ are respectively the lower and upper bounds on z, n_c is the number of collocation points, n_e is the number of finite elements, and element coupling matrices (arising from the continuity equations) are given by

$$\overline{G} = \begin{bmatrix} I & 0 & \dots & 0 \end{bmatrix}, \quad \underline{G} = \begin{bmatrix} 0 & \dots & 0 & -I & 0 & 0 \end{bmatrix}. \tag{7.28}$$

The coupling constraints $\underline{G}z_{i-1} + \overline{G}z_i{=}0$ link individual finite elements in time. It is important to note that only the state variables are temporally coupled between elements, not the algebraic variables. In other words, the $-I$ block in $\underline{G}$ corresponds only to x_{i,n_c}. Therefore, the dimension of these constraints, (i.e., the number of rows in $\underline{G}$ and $\overline{G}$) is dependent on the number of state variables only. It is this property that will be exploited to decompose the problem and develop an efficient parallel solution approach.

7.4 Parallel Solution of the Dynamic Optimization Problem

Introducing decoupling variables, $q_i{=}x_{i,n_c}$, $i{=}1..n_e{-}1$, we may rewrite the NLP resulting from collocation as

$$\min_{z} \ f(z) \tag{7.29a}$$

$$\text{s.t.} \ c(z,q) = 0 \tag{7.29b}$$

$$\underline{z} \leq z \leq \overline{z} \tag{7.29c}$$

where $\underline{z}$ and $\overline{z}$ are respectively the lower and upper bounds on z, $z^T = [z_1^T, \ldots, z_{n_e}^T]$, $q^T = [q_1^T, \ldots, q_{n_e-1}]$, and

$$c(z,q) = \begin{bmatrix} \overline{G} z_1 - x_0 \\ R(z_1) \\ \underline{G} z_1 + q_1 \\ \overline{G} z_2 - q_1 \\ R(z_2) \\ \underline{G} z_2 + q_2 \\ \vdots \\ \overline{G} z_{n_e} - q_{n_e-1} \\ R(z_{n_e}) \end{bmatrix}. \tag{7.30}$$

Solution of this large-scale nonlinear programming problem is possible with a number of potential algorithms. The dominant cost of an interior-point algorithm is the solution of the linear KKT system at each iteration to find the full step in the primal and dual variables. This linear system can be solved with direct factorization methods appropriate for symmetric indefinite systems, however, the structure of the optimal control problem induces structure in the KKT system that can be exploited with a parallel decomposition algorithm. This linear system can be decomposed by selecting break-points in time and performing a Schur-complement decomposition with respect to the coupling constraints (or corresponding multipliers) and variables.

Here, we describe our decomposition algorithm assuming that decoupling variables q and coupling constraints are added between each finite element. This gives a system where each finite element represents one decomposed block. However, our actual implementation is able to decompose the problem with multiple finite elements per block, where decoupling variables and coupling constraints are not added between every finite element. For example, a problem with 128 finite elements can be separated into two blocks of 64 finite elements each, 4 blocks of 32 finite elements each, and so forth. The number of blocks formed should be determined by the number of available processors.

The linear KKT system solved at each iteration of the interior-point optimization algorithm (specified in (7.14)) can be written in the following block-bordered structure. Again, for simplicity of notation, the structure is written with a decoupling variables introduced between every finite element.

$$\begin{bmatrix} K_1 & & & & A_1^T \\ & K_2 & & & A_2^T \\ & & \ddots & & \vdots \\ & & & K_{n_e} & A_{n_e}^T \\ A_1 & A_2 & \dots & A_{n_e} & Q \end{bmatrix} \begin{bmatrix} \Delta v_1 \\ \Delta v_2 \\ \vdots \\ \Delta v_{n_e} \\ \Delta v_s \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{n_e} \\ r_s \end{bmatrix} \tag{7.31}$$

where

$$K_i = \begin{bmatrix} H_i + \delta_H I & \overline{G}^T & \nabla_{z_i} R(z_i) \\ \overline{G} & -\delta_c I & \\ \nabla_{z_i} R(z_i)^T & & -\delta_c I \end{bmatrix} \forall\ i = 1, ..., n_e, \tag{7.32}$$

$$\Delta v_i = \begin{bmatrix} \Delta z_i \\ \Delta \lambda_{\overline{G},i} \\ \Delta \lambda_{R,i} \end{bmatrix},\ r_i = \begin{bmatrix} -\tilde{r}_{z_i} \\ -\overline{G} z_i + q_{i-1} \\ -R_i \end{bmatrix} \forall\ i = 1, ..., n_e, \tag{7.33}$$

$$A_1 = \begin{bmatrix} 0 & 0 & 0 \\ \underline{G} & 0 & 0 \\ 0 & 0 & 0 \\ & \vdots & \end{bmatrix} \tag{7.34}$$

$$A_i = \begin{bmatrix} 0 & 0 & 0 \\ & \vdots & \\ 0 & -I & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \underline{G} & 0 & \\ 0 & 0 & 0 \\ & \vdots & \end{bmatrix} \forall\ i = 2, ..., n_e - 1 \tag{7.35}$$

$$A_{n_e} = \begin{bmatrix} 0 & 0 & 0 \\ & \vdots & \\ 0 & 0 & 0 \\ 0 & -I & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{7.36}$$

$$Q = \begin{bmatrix} \delta_H I & I & & & & & \\ I & -\delta_c I & & & & & \\ & & \delta_H I & I & & & \\ & & I & -\delta_c I & & & \\ & & & & \ddots & & \\ & & & & & \delta_H I & I \\ & & & & & I & -\delta_c I \end{bmatrix}, \tag{7.37}$$

$$\Delta v_s = \begin{bmatrix} \Delta q_1 \\ \Delta \lambda_{\underline{G},1} \\ \vdots \\ \Delta q_{n_e-1} \\ \Delta \lambda_{\underline{G},n_e-1} \end{bmatrix}, \text{ and } r_s = \begin{bmatrix} -\nabla_q \tilde{r}_1 \\ -\underline{G} z_1 - q_1 \\ \vdots \\ -\nabla_q \tilde{r}_{n_e-1} \\ -\underline{G} z_{n_e-1} - q_{n_e-1} \end{bmatrix}. \tag{7.38}$$

Here, H_i is the modified Hessian described in (7.10), and δ_H and δ_c may be zero or positive depending on the need of the algorithm to handle non-convexity and/or singlularity in the Jacobian. The $\Delta\nu_i$ vectors include the primal and dual variables for element i, and $\Delta\nu_s$ contains the dual variables for the coupling constraints. In this permutation, the coupling constraints, (i.e., the Jacobian matrices $\underline{G}$ and $\overline{G}$), and their corresponding dual variables have been permuted to the borders of the KKT system.

The step in the variables v_s can be decoupled from the remaining variables by eliminating the A_i matrices, resulting in the following Schur-complement decomposition,

$$\left[Q - \sum_i A_i K_i^{-1} A_i^T\right] \Delta\nu_s = r_s - \sum_i A_i K_i^{-1} r_i. \tag{7.39}$$

This decomposition allows solution of the KKT system using the following algorithm.

Algorithm: Schur-Complement Solve of KKT System

```
1: for each $i$ in $1, ..., n_e$
    1.1: factor $K_i$ (using MA27 from Harwell Subroutine Library)
2: Initialize $S$ by letting $S = Q$ (shown in (7.37))
3: let $r_{sc} = r_s$
4: for each $i$ in $1, ..., n_e$
    4.1: for each nonzero column $j$ in $A_i^T$
        4.1.1: solve the system $K_i s_i^{<j>} = [A_i^T]^{<j>}$ for $s_i^{<j>}$
        4.1.2: let $S^{<j>} = S^{<j>} - A_i s_i^{<j>}$
    4.2: solve the system $K_i p_i = r_i$ for $p_i$
    4.3: let $r_{sc} = r_{sc} - A_i p_i$
5: solve $S \Delta\nu_s = r_{sc}$ for $\Delta\nu_s$
6: for each $i$ in $1, ..., n_e$
    6.1: solve $K_i \Delta\nu_i = r_i - A_i^T \Delta\nu_s$ for $\Delta\nu_i$
```

There are several levels of parallelism that can be exploited in this algorithm. If there is one processor available for each element, then Steps 1, 4, and 6 can all be parallelized by utilizing one processor for each i in n_e. Furthermore, if more processors are available, individual column backsolves in Step 4.1 can be parallelized. In the traditional Schur-complement decomposition approach, the number of columns in A_i^T is typically dependent on the overall number of coupling or first-stage variables. However, with this problem structure, the number of nonzero columns in A_i^T is dependent on the number of state variables only. Therefore, when solving the system in parallel, the size of the Schur-complement grows with the number of processors, but the number of backsolves required for Step 4.1 does not.

In previous work we have shown excellent parallel scalability using this strategy for problems with complicating variables (Zavala et al., 2008; Zhu et al., 2010) where the size of the Schur-complement is determined by the number of complicating variables only. In the approach described here, the size of the Schur-complement increases with the number of states and the number of processors used. As the size of the Schur-complement grows, the increased cost of solving the Schur-complement system (Step 5) will erode parallel speedup. In general, the Schur-complement is dense and the cost of solution with a dense linear solver would increase cubically with the size of the Schur-complement. However, due to the block structure induced on the A_k blocks by the collocation and decomposition, the Schur-complement here can be quite sparse. Figure 7.1 shows the block structure of the Schur-complement formed using our approach. While the dimension of the Schur-complement grows quadratically with the number of A_k blocks and is equal to $(2N-2)^2$, where N is the number of blocks, the number of nonzero blocks in the Schur-complement only grows linearly and is equal to $4+6(N-2)$. Note that N must be greater or equal to 2 for this decomposition to be utilized. The Schur-complement can be quite sparse when a large number of processors is used, and using efficient sparse linear solvers can dramatically reduce the computational time required to solve the Schur-complement, improving parallel scalability. In this work we use the sparse linear solver MA27 (HSL (2011)).

7.5 Implementation

In this section we make use of the JModelica.org (Åkesson et al., 2010) modeling framework that is based on Modelica, Optimica, and Python, to transform high-level descriptions of dynamic optimization problems into algebraic nonlinear programming problems through a direct collocation approach. JModelica.org is a comprehensive

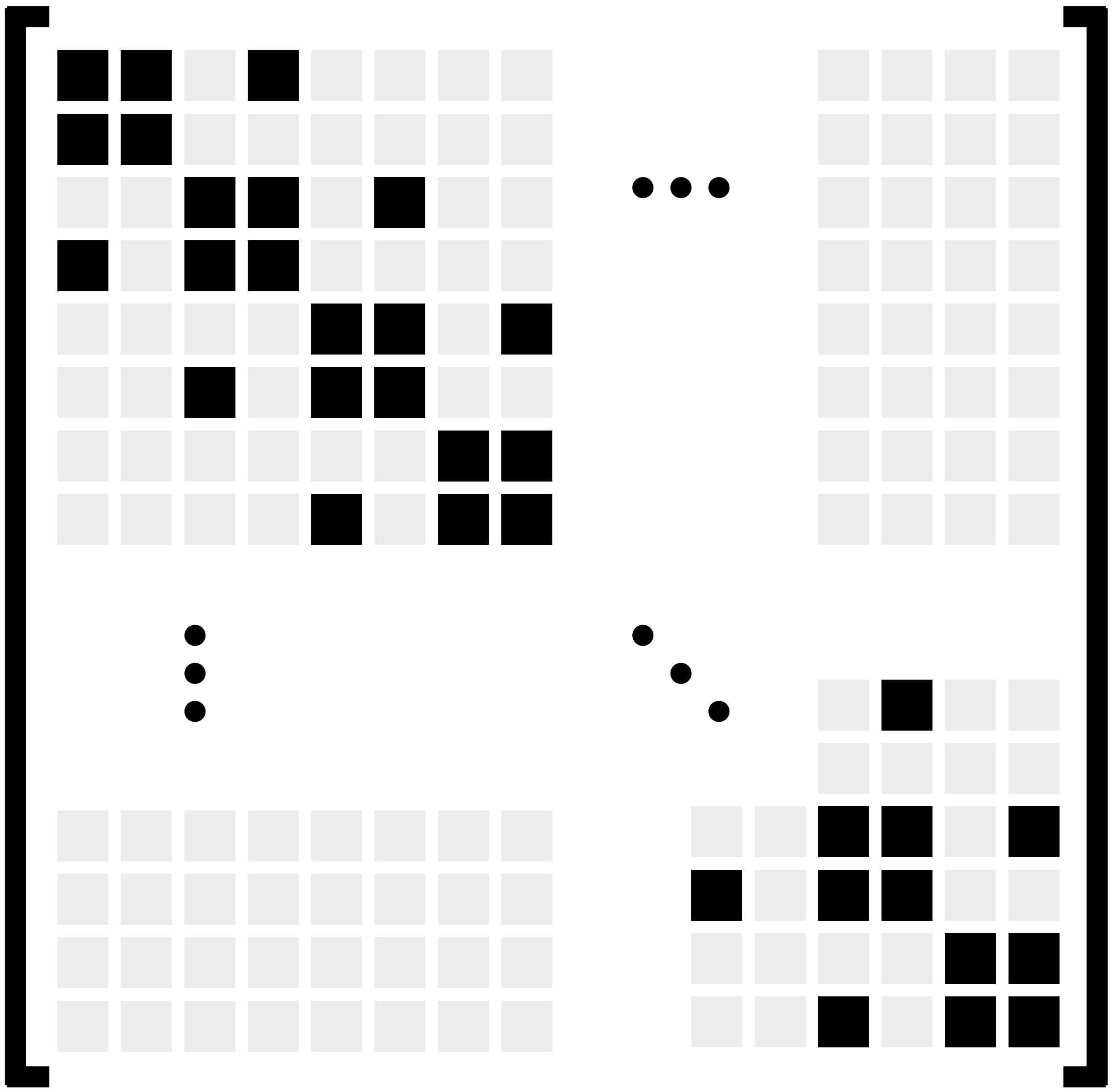

Figure 7.1: The Schur-complement sparsity pattern. The dark boxes represent blocks that can be dense and contain nonzeros, and the light boxes represent blocks that can only contain zeros.

modeling and optimization tool for large-scale dynamic optimization problems. The platform employs compiler technology, symbolic manipulation, and code generation to transform high-level Modelica and Optimica descriptions into efficient executables suitable for linking with numerical solvers. In addition, XML files containing DAE-constrained optimization problems can be generated from Modelica and Optimica descriptions. We interface with the open-source symbolic framework for automatic differentiation and optimal control, CasADi (Andersson et al., 2012), for efficient automatic differentiation. CasADi supports import of model XML files compliant with the format used by JModelica.org, which enables a seamless integration between the tools, (Andersson et al., 2011).

The interior-point algorithm is implemented in C++ using an object-oriented design where the core interior-point algorithm is independent of the specific problem representation and linear algebra routines. This design eases the development of custom decomposition approaches to exploit specific problem structure since the fundamental algorithm does not require change. This design also allows for straightforward parallelization since only the interfaces to the linear algebra routines are exposed to the fundamental algorithm, and the underlying operations function identically whether executed in serial or parallel. The algorithm requires several vector, vector-vector, and matrix-vector linear algebra operations (e.g. dot product, norms, matrix-vector multiplication), and in the parallel implementation these operations are parallelized using MPI routines.

7.6 Benchmark Problem

To demonstrate the performance characteristics of our algorithm we solve a problem to find the optimal control profile for the start-up of a combined cycle power plant. The power plant model is encoded in the object-oriented modeling language Modelica, and is based on a library consisting of model classes representing the model components. The model object diagram is shown in Figure 7.2. The key limiting factor with regards to start-up speed is the thermal stress in the steam turbine. The stress in the turbine axis is proportional to the temperature gradient, which in turn results when hot steam is exposed to the cold axis surface. Apart from the heat

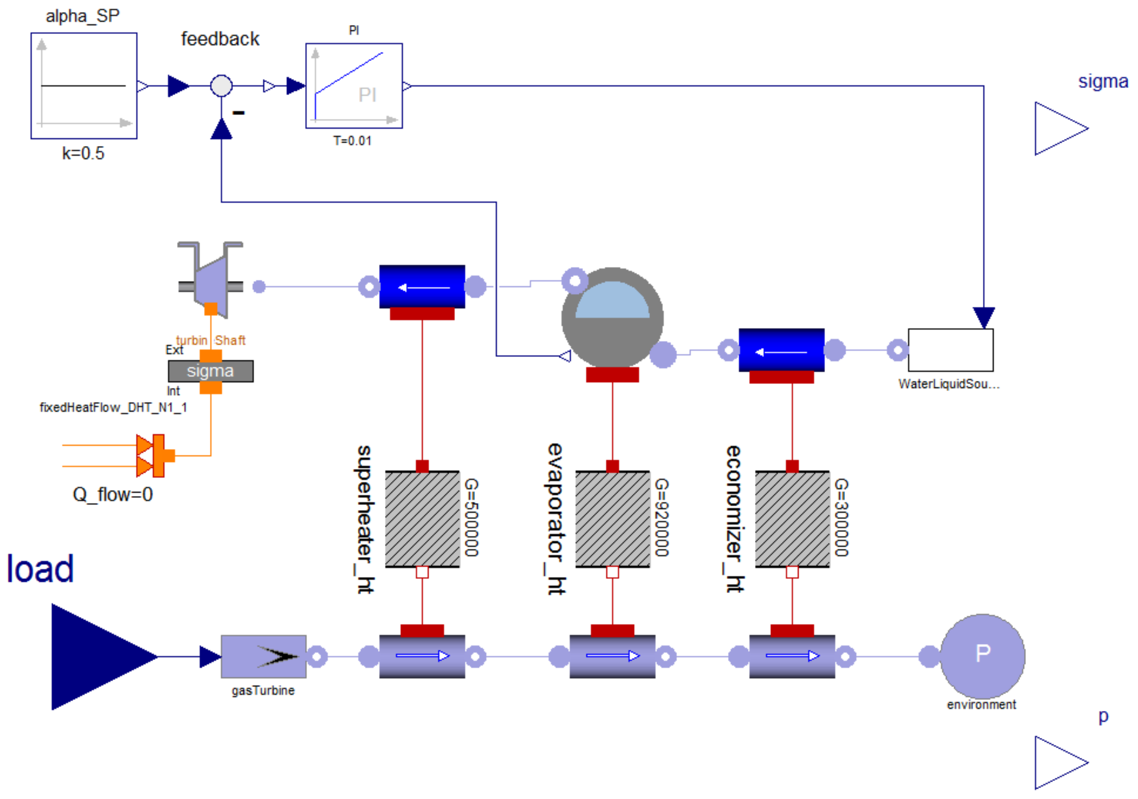

Figure 7.2: Object diagram of the combined cycle power plant.

transfer in the steam turbine axis, the main model elements are the boiler, an economizer and a super heater. The control input of the model is the load of the gas turbine. The model has ten states, 127 algebraic variables and one control input.

The optimal control problem seeks to determine a trajectory for the gas turbine load which drives the pressure in the boiler to a target value, corresponding to full production, while respecting a bound on the thermal stress in the steam turbine axis. In addition, there is a rate limit on the control input. For details on the models and the optimal control formulation, we refer to Casella et al. (2011).

7.7 Performance Results

The computational cost of the approach described in Section 7.4 is a function of the number of state variables, (i.e. the dimension of the coupling constraints), and the number of processors used in the decomposition. As we increase the number of processors, we have the potential for greater parallelization, however, the size of the Schur-complement (and hence the cost of Step 5) also increases. Increased parallelization also adds computational time required for inter-processor communication. Depending upon problem size, this additional communication burden can be significant.

We test the speedup of our algorithm on 3 test problems of different sizes with 64, 128, and 256 finite elements. Figure 7.3 shows the speedup of our parallel decomposition code when compared with the same decomposition performed in serial. In this example, a speedup of over 50 times was possible for the largest problem. Figure 7.4 shows the speedup for our parallel decomposition when compared with the serial full-space approach where the entire KKT system is solved directly with MA27. Here, we see that our parallel decomposition is still capable of appreciable speedup (over 50 times faster for the largest problem). Note that the speedups comparing the decomposition and full-space algorithms are per iteration of the interior-point algorithm rather than for the overall runtime. Recall that our approach introduces additional copuling variables to decompose the problem. To provide a fair comparison, we do not introduce these additional variables hwen solving with the full-space approach. Speedup per iteration is shown because the full-space and decomposition algorithms are solving different systems and may therefore take slightly different steps, which may require a different but comparable number of steps (and therefore interior-point iterations). In our experience neither algorithm consistently requires fewer iterations.

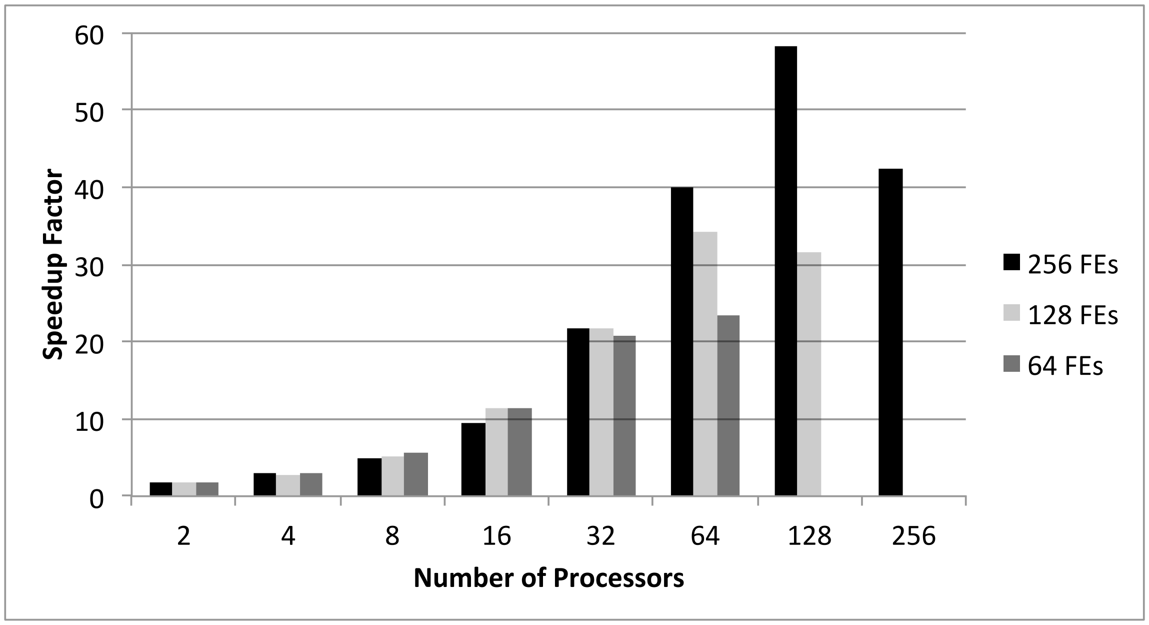

Figure 7.3: Speedups comparing the parallel Schur-complement linear solver with the serial Schur-complement linear solver for problems with 64, 128, and 256 finite elements.

While significant speedup is observed, as expected, the speedup of our parallel algorithm deteriorates when we increase to larger numbers of processors. This is due to two aspects. First, as the number of processors is increased, the computational time required by each processor decreases, while the communication overhead increases. For larger problems, where the computational burden per processor is higher, this effect is reduced. For this reason, we expect that our algorithm would demonstrate better speedup on larger problems (for the same number of state variables). Second, the time required to solve the Schur-complement (Step 5) increases with the number of blocks, and, while this time remains small, it makes an increasingly significant contribution to the solution time.

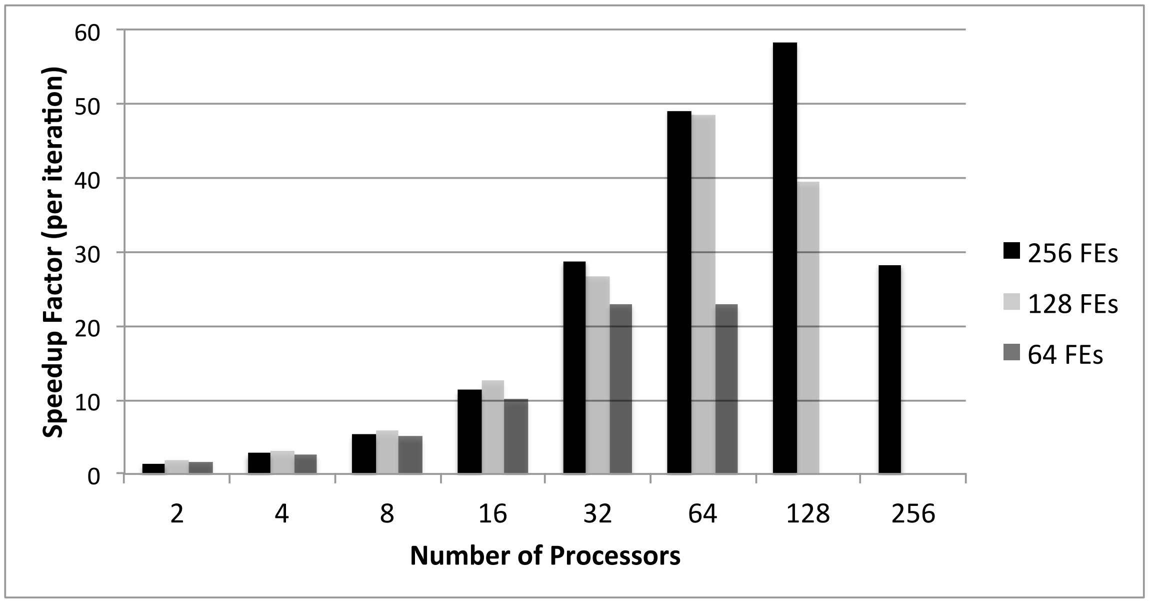

Figure 7.4: Speedups comparing the parallel Schur-complement linear solver with the full-space linear solver MA27 for problems with 64, 128, and 256 finite elements.

Table 7.1 details the times per iteration required in the computationally expensive steps of the linear solve. Both the serial and parallel times are shown for our test problem with varying numbers of finite elements and blocks. Increasing the number of blocks makes each individual block smaller, and factorizing and performing backsolves with these smaller blocks can be significantly faster. This is why as the number of blocks increases the times required for Steps 1 and 6 can actually decrease even when performing these calculations in serial. However, while the number of operations in Step 4 stays the same for each block, increasing the number of blocks increases the total number of operations, so even though individual backsolves may be faster, the time required for the serial algorithm increases. For the parallel algorithm, each block is distributed on separate processors so the time required for Step 4 does not increase until the inter-processor communication overhead becomes significant. This table also shows the increase in computational time required to solve the Schur-complement. For example, for the 256 finite element case Step 5 requires less than 0.005% of the linear solver time when using 2 blocks but more than 33% of the linear solver time when using 256 blocks.

Schur-Complement Times									
# of FEs	# of Blocks	Step 1		Step 4		Step 5		Step 6	
		Serial SC	Parallel SC	Serial SC	Parallel SC	Serial SC	Parallel SC	Serial SC	Parallel SC
64	2	0.35	0.197	0.10	0.05	7.1E-05	5.9E-05	1.2E-02	4.9E-03
	4	0.37	0.115	0.13	0.05	1.6E-04	1.5E-04	1.1E-02	3.0E-03
	8	0.35	0.058	0.14	0.03	3.2E-04	3.2E-04	1.1E-02	1.6E-03
	16	0.34	0.030	0.16	0.01	6.6E-04	6.7E-04	1.0E-02	5.8E-04
	32	0.22	0.010	0.17	0.01	1.4E-03	1.4E-03	8.5E-03	2.7E-04
	64	0.21	0.005	0.23	0.01	3.0E-03	3.1E-03	8.0E-03	3.0E-04
128	2	0.80	0.472	0.28	0.15	6.5E-05	7.6E-05	2.8E-02	1.4E-02
	4	0.83	0.280	0.31	0.13	1.7E-04	1.7E-04	2.5E-02	7.8E-03
	8	0.75	0.126	0.34	0.09	3.5E-04	3.7E-04	2.3E-02	4.4E-03
	16	0.73	0.062	0.37	0.03	7.3E-04	7.7E-04	2.2E-02	1.8E-03
	32	0.61	0.029	0.37	0.01	1.4E-03	1.4E-03	2.0E-02	7.1E-04
	64	0.39	0.009	0.45	0.01	2.9E-03	2.9E-03	1.7E-02	4.4E-04
	128	0.26	0.001	0.69	0.01	6.2E-03	5.4E-03	1.5E-02	7.6E-04
256	2	3.05	1.853	0.74	0.41	1.4E-04	1.1E-04	6.9E-02	3.5E-02
	4	2.28	0.754	0.90	0.34	1.6E-04	2.1E-04	6.1E-02	1.8E-02
	8	2.00	0.361	0.79	0.21	3.5E-04	3.5E-04	5.2E-02	9.6E-03
	16	1.78	0.174	0.71	0.09	6.6E-04	7.3E-04	4.6E-02	4.4E-03
	32	1.43	0.064	0.79	0.04	1.4E-03	1.5E-03	4.2E-02	2.4E-03
	64	1.40	0.034	1.00	0.02	2.9E-03	3.0E-03	4.1E-02	1.0E-03
	128	1.21	0.014	1.79	0.02	7.8E-03	8.1E-03	3.7E-02	1.2E-03
	256	0.84	0.005	3.66	0.03	1.9E-02	2.2E-02	3.4E-02	2.1E-03

Table 7.1: The table shows in seconds the average time per iteration required in the steps of the Schur-complement algorithm for the example problem as the number of finite elements and blocks is changed. Both serial and parallel times are shown.

All timing results were obtained using the Red Mesa supercomputing cluster at Sandia National Lab. This cluster is made up of computing nodes, each with two, 2.93 GHz quad-core, Nehalem X5570-processors, giving 8 computing cores per node. Each node has 12 GB of DDR3 RAM. For the results shown in Figures 7.3 and 7.4 only one computing core was used per computing node. However, computing clusters with distributed memory are typically much more expensive and less common than the shared memory architectures of the standard desktop computer. Computer manufactures are increasing the number of computing cores available on the standard desktop computer, and it is important to understand the performance that can be expected on these common computer architectures. Figure 7.5 compares the speedups achieved using distributed memory and shared memory architectures. The shared memory speedups come from solving the combined cycle powerplant start-up problem with 256 finite elements using an increasing number of computing cores on a single computing node. Note that as the number of processors increases, the speedup does not suffer.

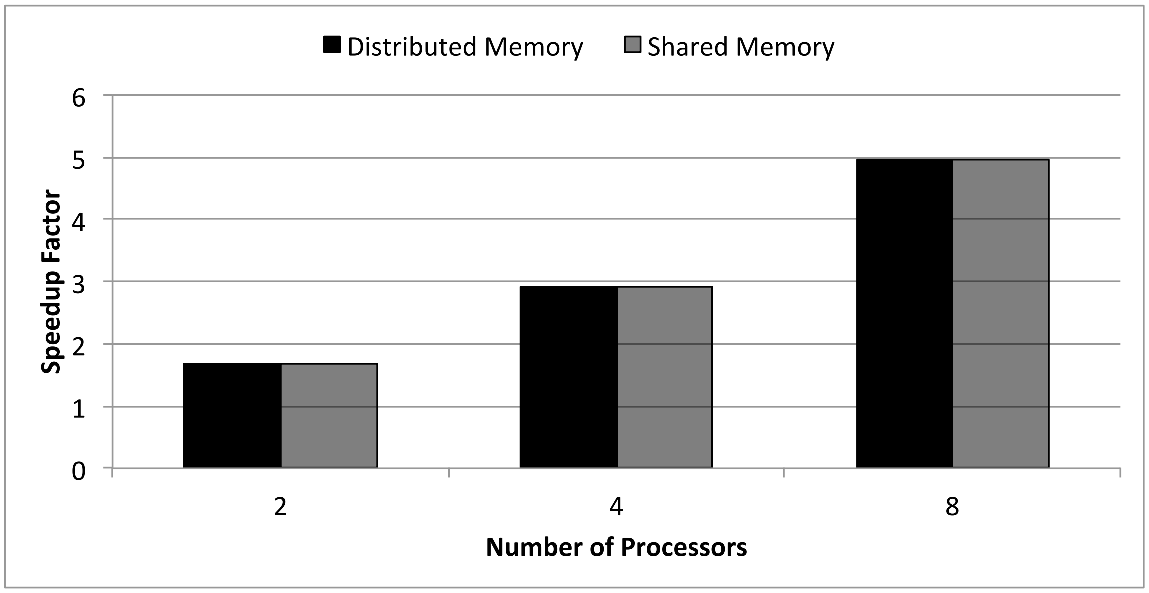

Figure 7.5: Speedups comparing shared-memory and distributed-memory architectures. These speedups compare our solution approach using the parallel Schur-complement algorithm over the serial Schur-complement algorithm on shared-memory and distributed-memory architectures.

7.8 Summary

This section presents a decomposition approach that is applicable for nonlinear dynamic optimization problems formulated using the simultaneous approach.

A Schur-complement algorithm is used for parallel solution of the linear systems resulting from an interior-point solution of these optimization problems. The dominant costs of the Schur-complement algorithm are Step 1 (factoring the K blocks), Step 4 (forming the Schur-complement), and Step 5 (solving the Schur-complement), however, Steps 1 and 4 can be efficiently parallelized, and for our test problem the computational cost of Step 5 remained small.

We interfaced our solution approach with the JModelica.org modeling framework that transforms high-level descriptions of dynamic optimization problems into algebraic nonlinear programming problems through a direct collocation approach (Åkesson et al., 2010). This modeling framework is integrated with CasADi (Andersson et al., 2012) to provide efficient automatic differentiation (Andersson et al., 2011).

For problems with few states and many algebraics, this solution approach has the potential for significant speedup. As the size of the Schur-complement increases (more states or processors), parallel speedup is eroded, and with an increasing number of processors the cost of inter-processor communication can become significant. This is especially true with relatively small problems that solve quickly, since the time required for communication can easily become a significant cost of the algorithm. Nevertheless, this approach can still be highly efficient. For blocks A_k of arbitrary structure, the Schur-complement may indeed be dense, but for dynamic optimization problems studied here, the structure of the A_k blocks is such that the resulting Schur-complement is both block structured and relatively sparse making it ideally suited for solution using sparse linear solvers. In this work, the efficient, serial sparse linear solver MA27 was used to solve the Schur-complement with promising results, but it could be possible to see further algorithm improvements if an efficient, parallel sparse linear solver was utilized.

The combined cycle power plant problem shows that this parallel decomposition approach can be coupled with a parallel interior-point algorithm for efficient solution of real optimal control problems. The case study displayed the potential for speedups of more than 50 over full-space approaches using distributed memory computing architectures. Additional results compared the speedup of our algorithm using shared memory and distributed memory computing architectures, and for the computing hardware used in our tests no differences were observed.

The nature of this algorithm lends itself to problems with a high number of algebraic variables compared to state variables. The infectious disease models that we have investigated thus far have a relatively low number of algebraic variables compared to state variables. Because of this, it is not likely that our Schur-complement decomposition algorithm would perform well for measles models, and therefore we have not investigated the use of this algorithm for this problem type.

8. SUMMARY, CONCLUSIONS, AND FUTURE WORK*

The development of infectious disease models remains important to provide scientists with tools to better understand disease dynamics and develop more effective control strategies. In this work we focused on the estimation of seasonally varying transmission parameters in infectious disease models from real measles case data.

In Section 1 we presented the motivation behind our research in parameter estimation for infectious disease models. The advancements in public health that are possible through the application of mathematical modeling and estimation techniques to infectious diseases can improve the lives of millions of people worldwide. Due to the availability of measles data, our research has focused on the estimation of transmission parameters for measles models using real measles data. Additionally, we investigated two parallel solution techniques that can be applied to very large problems that would be intractable using serial approaches.

*Part of this section is reprinted with permission from "'Exploiting Modern Computing Architectures for Efficient Large-Scale Nonlinear Programming" by Zhu, Y., Word, D., Siirola, J., and Laird, C.D., 2009. In: R.M. de Brito Alves, C.A.O. do Nascimento and E.C. Biscaia, Jr., Eds., Computer Aided Chemical Engineering, Copyright 2010 by Elsevier.

Part of this section is reprinted with permission from "Estimation of seasonal transmission parameters in childhood infectious disease using a stochastic continuous time model" by Word, D.P., Young, J., Cummings, D., and Laird, C.D., 2010. In: S. Pierucci and G. Buzi Ferraris, Eds., Computer Aided Chemical Engineering, Copyright 2010 by Elsevier.

Part of this section is reprinted with permission from "A Nonlinear Programming Approach for Estimation of Transmission Parameters in Childhood Infectious Disease Using a Continuous Time Model" by Word, D.P., Cummings, D.A.T., Burke, D.S., Iamsirithaworn, S., and Laird, C.D., 2012. Journal of the Royal Society Interface, Copyright 2012 by The Royal Society.

Part of this section is reprinted with permission from "A Progressive Hedging Approach for Parameter Estimation of Stochastic Nonlinear Programs" by Word, D.P., Watson, J.P., Woodruff, D., and Laird, C.D., 2012. Proceedings of PSE2012, Singapore, Copyright 2012 by Elsevier B.V.

Part of this section is reprinted with permission from "Interior-Point Methods for Estimating Seasonal Parameters in Discrete-Time Infectious Disease Models" by Word, D.P., Young, J.K., Cummings, D.A.T., Iamsirithaworn, S., and Laird, C.D., 2013. Submitted to PLOS One.

Part of this section is reprinted with permission from "Efficient Parallel Solution of Large-Scale Nonlinear Dynamic Optimization Problems" by Word, D.P., Kang, J., Akesson, J., and Laird, C.D., 2013. Submited to Computation Optimization and Applications.

A significant amount of work has been conducted in the epidemiology community on the modeling and estimation of infectious disease dynamics. Section 2 presents a review of several decades of research in this field and lays the foundation from which our modeling and estimation approaches grew. This review explains the premise of compartment-based modeling and details research in several discrete-time and continuous-time compartment based models that include a number of different compartments. Several estimation approaches are described that focus primarily on the estimation of seasonally varying transmission parameters. Efficiency of these techniques vary widely with some requiring only minutes and others requiring many hours, and results from several of the studies referenced in this section are compared against our results.

Section 3 describes the basic approach of the primal-dual interior-point methods for nonlinear programming that are used throughout this research. This topic is presented here since a basic understanding of these methods is important in the development of our solution approaches. These techniques have proven to be highly effective and efficient for solving many very large-scale and complex problems across many industries. Furthermore, these techniques can be easily extended to allow for parallel solution techniques.

Multiple discrete-time models were presented in Section 4 to demonstrate the flexibility inherent in large-scale nonlinear programming techniques and the ability of these techniques to efficiently estimate transmission parameters in multiple disease models using measles case count data. We demonstrated this efficiency and flexibility using three model formulations and performing estimation using three real data sets. In all cases, including for time-series data sets of up to 20 years, we were able to perform the estimations in less than 6 seconds.

We validated our estimation approach using simulated case data where parameters were known. We also compared our estimates using case data from London with estimates found in the literature, and found our results to be consistent with prior findings (He et al., 2010; Hooker et al., 2011). While our estimates were consistent with the literature, they are significantly higher than the published estimates of R_0 (Anderson and May, 1991). This deviation may be due to dependency between model parameters (as shown in the confidence regions in Figures 4.7-4.9), or because of the discrete-time approximation. Estimation results presented in Section 5 and appearing in Word et al. (2012) which are based on a continuous-time model do not show this deviation from estimates of R_0.

Using real measles case data from both New York City and Bangkok, we performed estimates using 3 discrete-time formulations that considered seasonality in the transmission parameter, the exponential parameter, and the introduction of new susceptibles. In all cases, the estimated seasonality showed correlation with school schedules. This is especially important given that the school schedules differ significantly for these two locations. The profile estimated using seasonal exponential parameters was practically identical to that estimated using a seasonal transmission parameter. This result might not be too surprising, but this does highlight that care must be taken when relating the estimated seasonality to particular system phenomena (e.g., contact rate).

The estimation results for the model with a seasonal weighting of the births also showed correlation to school holidays. Here, instead of assuming that new susceptibles always entered the population uniformly throughout the year, the model was formulated so that susceptibles could enter the population in any seasonal pattern. These estimation results show that all new susceptibles were introduced to the population immediately following long school holidays to best capture the dynamics observed in reported measles cases. Since all births clearly do not actually occur at this time, this result is consistent with the idea that the susceptible children impact observed measles dynamics when they enter the school population.

Successful estimation of parameters in dynamic models for childhood infectious diseases from time-series data presents several challenges. Typically, reported cases (the incidence) are the only available data, while there is little information about the susceptible population. Therefore, approaches must simultaneously estimate the prevalence and the unknown susceptible states. Furthermore, the case data is often significantly under-reported, the reporting interval is often longer than the serial interval of the disease, and the models are highly nonlinear. Section 5 presented a nonlinear programming approach for estimating the unknown states and the seasonal transmission parameter using a continuous-time model with both measurement and model noise.

Continuous time formulations offer several advantages over discrete-time formulations for estimation of infectious disease models. Data can be handled in its native form regardless of the reporting interval. This was demonstrated by using biweekly reported data from London and monthly reported data from New York City and Bangkok. Using data in its native form is a significant advantage for diseases with short serial intervals where it would be unreasonable to have data reported at the same interval.

The estimated transmission profiles from all three cities show strong correlation with school holidays despite the very different holiday schedules seen between London, New York City, and Bangkok. For Bangkok and NYC there was a lag observed in the estimated transmission profiles that showed the drop in transmission as occurring after the holiday had begun. This is probably due to a lag in reporting causing cases to be reported well after their occurrence and the incubation period of measles causing cases to be observed after the start of the holiday even though the infection occurred before the holiday.

The estimation approach presented here is highly efficient. Recent work by Hooker et al. (2011) solves a similar problem formulation in approximately 2 hours, the MCMC estimation performed by Cauchemez and Ferguson (2008) required approximately 20 hours per run, and the plug-and-play method of He et al. (2010) requires approximately 5 hours. None of our estimations take longer than 5 minutes. The efficiency of this fully simultaneous approach opens the door to explore many more model structures efficiently and provides a framework that is scalable to large spatially distributed estimations.

Several recent publications have reported estimated values of the seasonal transmission parameter, and corresponding R_0 values, that are higher than estimates provided in Anderson and May (1991). For example, the reported estimates of He et al. (2010) for London give an R_0 of 57 with 95% confidence intervals of 37 and 60. There is significant complexity in finding R_0 values while considering seasonal transmission rates, and it is difficult to compare results arising from different model structures. Using the approximate relationship $R_0{=}\bar{\beta}(t)/\gamma$, we estimate $R_0{=}13.3$ in London with 95% confidence intervals of 12.1 and 14.3. Our estimates for New York City ($R_0{=}9.1$) and Bangkok ($R_0{=}17.9$) also give values for R_0 that appear consistent with values reported for measles in Anderson and May for other cities (Anderson and May, 1991) and with values approximated using the average age of infection.

The nonlinear programming approach for estimating infections disease models that was used in Sections 4 and 5 for estimating continuous-time infectious disease models is reliable, flexible, and efficient. Solutions to the nonlinear programming problems were possible with a general initialization strategy, and effective parameter estimates are possible, even in the face of challenging sets of data that contain missing years, severe under-reporting, and significant noise. It is straightforward to switch between diseases with different serial intervals or data sets with different reporting intervals. The approach is independent of model specifics. For example, it would be straightforward to add additional compartments to the model, such as adding an E compartment to make an SEIR model that would account for individuals that have been exposed to a disease but cannot yet infect susceptibles. One could also add a compartment to account for portions of the population that were vaccinated against a disease. Furthermore, the approach is highly efficient, making it appropriate for much larger problem formulations, or for rapid exploration and comparison of multiple model structures.

Using this flexible framework, we propose to address two important advances in future work. First, standard assumptions with the SIR model give exponential distributions in age-dependence of cases. This is contrary to the age-distributed case data we have for these locations. We propose to develop an age and time discretized model to estimate seasonal age-dependent transmission parameters using this approach. Second, while this work focused on estimating transmission parameters for individual large cities, another interesting problem for health officials is looking at a spatial model of disease spread. For accurate estimation of disease dynamics in small cities where fadeout is observed, information is needed regarding the transmission of the disease from a large city where the disease is endemic to the small city. The approach described in this paper is appropriate for estimation of large-scale, spatially distributed, nonlinear differential equations models and will be a subject of future research.

However, despite the high efficiency of these solution approaches, the future problems that we wish to address are very large and it is still straightforward to formulate estimation problems that are intractably large for serial solution approaches. With the focus of computer manufacturers on development of parallel computing architectures, it is imperative to develop solution algorithms that can exploit the unique capabilities of parallel computing so that larger problems can be tackled than what is currently possible.

In Section 6 we presented an equivalence between nonlinear programming parameter estimation problems and stochastic programming problems that allows us to use stochastic programming algorithms to estimate parameters. Specifically, this section demonstrates the use of the Progressive Hedging (PH) algorithm of Rockafellar and Wets (1991) to estimate transmission parameters for a continuous-time measles model using measles case data from 60 cities in England and Wales.

The estimated values of the seasonal transmission parameter found using the extensive form and the PH approach yield nearly identical solutions. This solution is also consistent with published values (Finkenstädt and Grenfell, 2000). What is equally important is that the PH algorithm can be easily implemented in parallel which can yield significant performance improvements. Not only can a solution be found much faster when using PH in parallel, but because the problem is distributed across multiple computers, memory requirements are much lower allowing much larger problems to be solved.

Further improvements will allow even greater speed-up for parallel PH. Currently, if even one scenario requires significantly longer to converge than the rest of the scenarios, a considerable number of computing nodes remain idle for extended periods of time. Asynchronous approaches being studied will reduce the number of idle processors by allowing the majority of computing nodes to continue with the next iteration of the algorithm using an altered update formula. Improved strategies for the selection of the algorithm parameter ρ could also further reduce solution times.

The PH algorithm decomposes problems by scenarios. This approach is suitable for problems that can be formulated as separate scenarios such as those with multiple data sets like the disease problems we have investigated. Another decomposition strategy is to decompose problems in time, and since our disease models are dynamic it is possible to decompose these problems in this way. In Section 7 we present a decomposition approach that is applicable to nonlinear dynamic optimization problems formulated using the simultaneous approach. A Schur-complement algorithm is used for parallel solution of the linear systems resulting from an interior-point solution of these optimization problems. While certain steps of the Schur-complement algorithm can be computationally expensive, some of these steps can be efficiently performed in parallel. The dominant serial cost of this algorithm is directly related to the number of states in the problem and the number of processors used to solve the problem.

For problems with few states and many algebraics, this solution approach has the potential for significant speedup, however, as the size of the Schur-complement increases (more states or processors), parallel speedup is eroded. Nevertheless, this approach can still be highly efficient. Because of structure induced in the problem through the collocation based discretization strategy used in the simultaneous approach, the resulting Schur-complement is both block structured and relatively sparse. The use of an efficient sparse linear solver dramatically decreases the time to solve the large Schur-complement and allows for significantly improved speedup compared to a dense linear solver. In this section, the efficient, serial sparse linear solver MA27 was used to solve the Schur-complement, but it could be possible to see further algorithm improvements if an efficient, parallel sparse linear solver was utilized.

A test problem shows that this parallel decomposition approach can be coupled with a parallel interior-point algorithm for efficient solution of real optimal control problems where the problem possesses a high ratio of algebraic to state variables. The case study displayed the same trade-off between increasing the number of processors and increased time for solving the Schur-complement as was observed in Laird et al. (2011), but the use of an efficient sparse linear solver keeps these times relatively small for our test problem. Additional results compared the speedup of our algorithm using shared memory and distributed memory computing architectures, but for the computing hardware used in our tests no differences were observed.

While this parallel algorithm showed significant performance improvements for the problems tested, it does not perform well for problems with a low ratio of algebraic to state variables. The infectious disease models that we have formulated have a low ratio of algebraic to state variables so we did not test this algorithm on the disease problems we have formulated.

Continued efforts will pursue the development of a true spatio-temporal model of infectious disease dynamics that includes unknown parameters describing transmission of disease between cities. This model will lead to a very large estimation problem that will require a parallel solution approach for efficient solution. The Progressive Hedging algorithm is one promising approach to solve this problem. This approach would allow the problem to be decomposed by cities so that each processor could look at a problem only as big as the estimation problem for a single city - a problem of tractable size.

Additionally, parallel solution approaches exploiting temporal decomposition are promising. Further research of this strategy could lead to an algorithm suitable even for problems with a low ratio of algebraic to state variables like our disease problems. It could also be promising to pursue algorithms that include scenario-based and time-based decomposition approaches to allow even further decomposition and utilization of parallel computing.

REFERENCES

Åkesson, J., Årzén, K., Gäfvert, M., Bergdahl, T., and Tummescheit, H. (2010). Modeling and optimization with Optimica and JModelica.org—languages and tools for solving large-scale dynamic optimization problem. *Computers and Chemical Engineering*, 34(11):1737–1749.

Amestoy, P., Duff, I., and L'Excellent, J. (2000). Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computer Methods in Applied Mechanics and Engineering*, 184(2):501–520.

Anderson, R. and May, R. (1991). *Infectious Diseases of Humans: Dynamics and Control*. Oxford University Press Inc., New York.

Andersson, J., Åkesson, J., Casella, F., and Diehl, M. (2011). Integration of casadi and jmodelica.org. In Claus, C., editor, *8th International Modelica Conference*, pages 218–231, Linkoping, Sweden. Linkoping University Electronic Press.

Andersson, J., Åkesson, J., and Diehl, M. (2012). Casadi: A symbolic package for automatic differentiation and optimal control. In Forth, S., Hovland, P., Phipps, E., Utke, J., and Walther, A., editors, *Recent Advances in Algorithmic Differentiation*, pages 297–307. Springer Berlin Heidelberg, Berlin.

Association, M. et al. (2007). *The Modelica Language Specification Version 3.0*. Modelica Association, Linkoping, Sweden.

Aster, R., Borchers, B., and Thurber, C. (2012). *Parameter Estimation and Inverse Problems*. Academic Press, New York.

Bailey, N. and Bailey, L. (1987). *The Mathematical Theory of Infectious Diseases.* A Charles Griffin Book, Hafner, New York, 2nd edition.

Bartlett, M. (1957). Measles periodicity and community size. *Journal of the Royal Statistical Society. Series A (General)*, 120(1):48–70.

Betts, J. (2010). *Practical Methods for Optimal Control and Estimation using Nonlinear Programming*, volume 19. Society for Industrial & Applied Mathematics, Philadelphia.

Betts, J. and Kolmanovsky, I. (2002). Practical methods for optimal control using nonlinear programming. *Applied Mechanics Reviews*, 55:B68.

Biegler, L. (2010). *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes.* SIAM, Philadelphia.

Biegler, L., Cervantes, A., and Wächter, A. (2002). Advances in simultaneous strategies for dynamic process optimization. *Chemical Engineering Science*, 57(4):575–593.

Biegler, L. and Grossmann, I. (2004). Retrospective on optimization. *Computers & Chemical Engineering*, 28(8):1169–1192.

Biegler, L. and Zavala, V. (2009). Large-scale nonlinear programming using ip: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575–582.

Biros, G. and Ghattas, O. (2005). Parallel lagrange-newton-krylov-schur methods for pde-constrained optimization. part i: The krylov-schur solver. *SIAM J. Sci. Comput.*, 27:687–713.

Bjornstad, O., Finkenstädt, B., and Grenfell, B. (2002). Dynamics of measles epidemics: Estimating scaling of transmission rates using a time series sir model. *Ecological Monographs*, 72:169–184.

Bobashev, G., Ellner, S., Nychka, D., and Grenfell, B. (2000). Reconstructing susceptible and recruitment dynamics from measles epidemic data. *Mathematical Population Studies*, 8(1):1–29.

Byrd, R., Hribar, M., and Nocedal, J. (1999). An interior point method for large scale nonlinear programming. *SIAM Journal of Optimization*, 9(4):887–900.

Byrd, R., Nocedal, J., and Waltz, R. (2006). KNITRO: An integrated package for nonlinear optimization. In di Pillo, G. and Roma, M., editors, *Large-Scale Nonlinear Optimization*, pages 35–59, New York. Springer Verlag.

Casella, F., Donida, F., and Åkesson, J. (2011). Object-oriented modeling and optimal control: A case study in power plant start-up. In Bittanti, S., Cenedese, A., and Zampieri, S., editors, *18th IFAC World Congress*, pages 9549–9554, Milano, Italy. Elsevier.

Cauchemez, S. and Ferguson, N. (2008). Likelihood-based estimation of continuous-time epidemic models from time-series data: application to measles transmission in London. *Journal of The Royal Society Interface*, 5(25):885.

Cervantes, A. and Biegler, L. (2000). A stable elemental decomposition for dynamic process optimization. *Journal of Computational and Applied Mathematics*, 120(1):41–57.

Cervantes, A., Wächter, A., Tütüncü, R., and Biegler, L. (2000). A reduced space interior point strategy for optimization of differential algebraic systems. *Computers & Chemical Engineering*, 24(1):39–51.

Cintrón-Arias, A., Banks, H., Capaldi, A., and Lloyd, A. (2009). A sensitivity matrix based methodology for inverse problem formulation. *Journal of Inverse and Ill-posed Problems*, 15:545–564.

COOPR (2008). Coopr: A common optimization python repository. `https://software.sandia.gov/coopr`. [Online; accessed 28-March-2013].

Daley, D. and Gani, J. (1999). *Epidemic Modelling: An Introduction*. Cambridge University Press, Cambridge.

DeMiguel, V. and Nogales, F. (2008). On decomposition methods for a class of partially separable nonlinear programs. *Mathematics of Operations Research*, 33(1):119–139.

Diehl, M., Bock, H., Schlöder, J., Findeisen, R., Nagy, Z., and Allgöwer, F. (2002). Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585.

Diekmann, O. and Heesterbeek, J. (2000). *Mathematical Epidemiology of Infectious Diseases*. John Wiley & Sons Ltd, New York.

Dietz, K. (1967). Epidemics and rumours: A survey. *Journal of the Royal Statistical Society*, 130(4):505–528.

Dietz, K. (1988). The first epidemic model: A historical note on p.d. en'ko. *Australian & New Zealand Journal of Statistics*, 30A(1):56–65.

Ellner, S., Bailey, B., Bobashev, G., Gallant, A., Grenfell, B., and Nychka, D. (1998). Noise and nonlinearity in measles epidemics: Combining mechanistic and statistical approaches to population modeling. *The American Naturalist*, 151(5):425–440.

Ferguson, N., Cummings, D., Cauchemez, S., Fraser, C., Riley, S., Meeyai, A., Iamsirithaworn, S., and Burke, D. (2005). Strategies for containing an emerging influenza pandemic in southeast asia. *Nature*, 437(7056):209–214.

Fine, P. and Clarkson, J. (1982). Measles in england and wales i: An analysis of factors underlying seasonal patterns. *International journal of Epidemiology*, 11(1):5–14.

Finkenstädt, B., Bjornstad, O., and Grenfell, B. (2002). A stochastic model for extinction and recurrence of epidemics: estimation and inference for measles outbreaks. *Biostatistics*, 3:493–510.

Finkenstädt, B. and Grenfell, B. (2000). Time series modelling of childhood diseases: a dynamical systems approach. *Journal of the Royal Statistical Society, Series C*, 49:187–205.

Forsgren, A., Gill, P., and Wright, M. (2002). Interior methods for nonlinear optimization. *SIAM Review*, 44(4):525–597.

Foundation, P. S. (1990). Python programming language. `http://www.python.org`. [Online; accessed 28-March-2013].

Fourer, R., Gay, D., and Kernighan, B. (1993). *AMPL: A Modeling Language for Mathematical Programming*. The Scientific Press (now an imprint of Boyd & Fraser Publishing Co.), Danvers, MA, USA.

Gallant, A. (1987). *Nonlinear Statistical Models*. Wiley, New York.

Glass, K., Xia, Y., and Grenfell, B. (2003). Interpreting time-series analyses for continuous-time biological models–measles as a case study. *Journal of Theoretical Biology*, 223(1):19–25.

Gomes, M., Gomes, J., and Paulo, A. (1999). Diphtheria, pertussis, and measles in portugal before and after mass vaccination: A time series analysis. *European Journal of Epidemiology*, 15(1):791–798.

Goulart, P., Kerrigan, E., and Ralph, D. (2008). Efficient robust optimization for robust control with constraints. *Mathematical Programming*, 114(1):115–147.

Gould, N., Orban, D., and Toint, P. (2004). Numerical methods for large-scale nonlinear optimization. Technical Report RAL-TR-2004-032, Central Laboratory of the Research Councils, Oxfordshire, England.

Greenhalgh, D. and Moneim, I. (2003). SIRS epidemic model and simulations using different types of seasonal contact rate. *Systems Analysis Modelling Simulation*, 43(5):573–600.

Grenfell, B. (2012). Pathogen population dynamics. http://www.zoo.cam.ac.uk/zoostaff/grenfell/measles.htm. [Online; accessed 28-March-2013].

Grenfell, B., Bjornstad, O., and Finkenstädt, B. (2002). Dynamics of measles epidemics: Scaling noise, determinism, and predictability with the tsir model. *Ecological Monographs*, 72:185–202.

Hamer, W. (1906). *Epidemic Disease in England.* The Milroy Lectures. Bedford Press, Bedfordbury, W.C.

Hart, W., Laird, C., Watson, J., and Woodruff, D. (2012). *Pyomo: Optimization Modeling in Python*, volume 67. Springer Verlag, New York.

Hart, W., Watson, J., and Woodruff, D. (2011). Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219–260.

Hartwich, A. and Marquardt, W. (2010). Dynamic optimization of the load change of a large-scale chemical plant by adaptive single shooting. *Computers & Chemical Engineering*, 34(11):1873–1889.

Hartwich, A., Stockmann, K., Terboven, C., Feuerriegel, S., and Marquardt, W. (2011). Parallel sensitivity analysis for efficient large-scale dynamic optimization. *Optimization and Engineering*, 12(4):489–508.

He, D., Ionides, E., and King, A. (2010). Plug-and-play inference for disease dynamics: measles in large and small populations as a case study. *Journal of The Royal Society Interface*, 7(43):271.

Herman, A. and Conway, B. (1996). Direct optimization using collocation based on high-order Gauss-Lobatto quadrature rules. *Journal of Guidance Control and Dynamics*, 19(3):592–599.

Hethcote, H. (2000). The mathematics of infectious diseases. *SIAM Review*, 42(4):599–653.

Hethcote, H. W. (1994). A thousand and one epidemic models. In Levin, S., editor, *Frontiers in mathematical biology*, number 1, pages 504–515. Springer, New York.

Hooker, G., Ellner, S., Roditi, L., and Earn, D. (2011). Parameterizing state–space models for infectious disease dynamics by generalized profiling: measles in Ontario. *Journal of The Royal Society Interface*, 8(60):961–974.

Houska, B., Ferreau, H., and Diehl, M. (2011). Acado toolkit–an open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3):298–312.

HSL (2011). *A collection of Fortran codes for large-scale scientific computation.* AEA Technology, Harwell, Oxfordshire, England. See http://www.hsl.rl.ac.uk.

Jandarov, R., Haran, M., Bjørnstad, O., and Grenfell, B. (2013). Emulating a gravity model to infer the spatiotemporal dynamics of an infectious disease. *arXiv preprint arXiv:1110.6451*.

Kang, J., Word, D., and Laird, C. (2013). An efficient interior-point decomposition algorithm for parallel solution of large-scale nonlinear problems with significant variable coupling. *Submitted to Computers & Chemical Engineering*.

Keeling, M. (1997). Modelling the persistence of measles. *Trends in Microbiology*, 5(12):513–518.

Keeling, M. and Eames, T. (2005). Networks and epidemic models. *Journal of the Royal Society Interface*, 2(1):295–307.

Kermack, W. and McKendrick, A. (1927). Contributions to the mathematical theory of epidemics – i. *Proceedings of the Royal Society of London Series A*, 115:700–721.

Kocak, S. and Akay, H. (2001). Parallel schur complement method for large-scale systems on distributed memory computers. *Applied Mathematical Modelling*, 25(10):873–886.

Laird, C. and Biegler, L. (2008). Large-scale nonlinear programming for multi-scenario optimization. In Bock, H., Kostina, E., Phu, H., and Rannacher, R., editors, *Modeling, Simulation and Optimization of Complex Processes*, pages 323–336, New York. Springer.

Laird, C., Biegler, L., van Bloemen Waanders, B., and Bartlett, R. (2005). Contamination source determination for water networks. *Journal of Water Resources Planning and Management*, 131(2):125–134.

Laird, C., Wong, A., and Akesson, J. (2011). Parallel solution of large-scale dynamic optimization problems. In Pistikopoulos, E., Georgiadis, M., and Kokossis, A., editors, *21st European Symposium on Computer Aided Process Engineering – ESCAPE*, volume 21, pages 813–817, Philadelphia. Elsevier B.V.

Lang, Y.-D. and Biegler, L. (2007). A software environment for simultaneous dynamic optimization. *Computers & Chemical Engineering*, 31(8):931–942.

Lee, Y., Nelder, J., and Pawitan, Y. (2006). *Generalized Linear Models with Random Effects: Unified Analysis via H-likelihood*, volume 106. CRC Press, Boca Raton.

Leibman, M., Edgar, T., and Lasdon, L. (1992). Efficient data reconciliation and estimation for dynamic processes using nonlinear programming techniques. *Computers & Chemical Engineering*, 16(10):963–986.

Leineweber, D., Bauer, I., Bock, H., and Schlöder, J. (2003). An efficient multiple shooting based reduced sqp strategy for large-scale dynamic process optimization. part 1: theoretical aspects. *Computers & Chemical Engineering*, 27(2):157–166.

Liu, W., Hethcote, H., and Levin, S. (1987). Dynamical behavior of epidemiological models with nonlinear incidence rates. *Journal of Mathematical Biology*, 25(4):359–380.

Lloyd, A. (2001). Realistic distributions of infectious periods in epidemic models: Changing patterns of persistence and dynamics. *Theoretical Population Biology*, 60(1):59–71.

Lotka, A. (1922). The stability of the normal age distribution. *Proceedings of the National Academy of Sciences of the USA*, 8:339–345.

Mattsson, S. and Söderlind, G. (1993). Index reduction in differential-algebraic equations using dummy derivatives. *SIAM Journal on Scientific Computing*, 14(3):677–692.

McKendrick, A. (1925). Applications of mathematics to medical problems. *Proceedings of the Edinburgh Mathematical Society*, 44:98–130.

Mollison, D. and Din, S. (1993). Deterministic and stochastic models for the seasonal variability of measles transmission. *Mathematical Biosciences*, 117(1):155–177.

Mulvey, J., Rosenbaum, D., and Shetty, B. (1997). Strategic financial risk management and operations research. *European Journal of Operational Research*, 97(1):1–16.

of Epidemiology, B. (1986). *Annual Epidemiological Surveillance Report: Measles*. Department of Disease Control, Ministry of Public Health, Bangkok, Thailand.

Petra, C. and Anitescu, M. (2012). A preconditioning technique for schur complement systems arising in stochastic optimization. *Computational Optimization and Applications*, 52(2):315–344.

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

Rao, C., Wright, S., and Rawlings, J. (1998). Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications*, 99(3):723–757.

Rockafellar, R. and Wets, R. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1):119–147.

Rooney, W. and Biegler, L. (2004). Design for model parameter uncertainty using nonlinear confidence regions. *AIChE Journal*, 47(8):1794–1804.

Rosenthal, R. (2012). *GAMS: A User's Guide.* GAMS Development Corporation, Washington D.C.

Ross, R. (1910). *The Prevention of Malaria.* E.P. Dutton & Company, New York.

Ruszczyński, A. (1993). Parallel decomposition of multistage stochastic programming problems. *Mathematical Programming*, 58(1):201–228.

Schenk, O. and Gärtner, K. (2004). Solving unsymmetric sparse systems of linear equations with pardiso. *Future Generation Computer Systems*, 20(3):475–487.

Schenk, O., Manguoglu, M., Sameh, A., Christen, M., and Sathe, M. (2009). Parallel scalable pde-constrained optimization: antenna identification in hyperthermia cancer treatment planning. *Computer Science-Research and Development*, 23(3):177–183.

Schenzle, D. (1984). An age-structured model of pre-and post-vaccination measles transmission. *Mathematical Medicine and Biology*, 1(2):169.

Scheu, H. and Marquardt, W. (2011). Sensitivity-based coordination in distributed model predictive control. *Journal of Process Control*, 21(5):715–728.

Scott, J. (2003). Parallel frontal solvers for large sparse linear systems. *ACM Transactions on Mathematical Software (TOMS)*, 29(4):395–417.

Seber, G. and Wild, C. (1989). *Nonlinear Regression*. Wiley, New York.

Soper, H. (1929). The interpretation of periodicity in disease prevalence. *Journal of the Royal Statistical Society*, 92(1):34–73.

Tanaka, R. and Martins, C. (2011). Parallel dynamic optimization of steel risers. *Journal of Offshore Mechanics and Arctic Engineering*, 133(1):011302–1–011302–9.

Tjoa, I. and Biegler, L. (1991). Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic equation systems. *Industrial & Engineering Chemistry Research*, 30(2):376–385.

van Bloemen Waanders, B., Bartlett, R., Biegler, L., and Laird, C. (2003). Nonlinear programming strategies for source detection of municipal water networks. Technical report, Sandia National Laboratories, Albuquerque.

Varziri, M., Poyton, A., McAuley, K., McLellan, P., and Ramsay, J. (2008). Selecting optimal weighting factors in ipda for parameter estimation in continuous-time dynamic models. *Computers & Chemical Engineering*, 32(12):3011–3022.

Wächter, A. (2002). *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. PhD thesis, Carnegie Mellon University, Pittsburgh.

Wächter, A. and Biegler, L. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

Waltz, R., Morales, J., Nocedal, J., and Orban, D. (2006). An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Mathematical Programming*, 107(3):391–408.

Watson, J. and Woodruff, D. (2011). Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science*, 8(4):355–370.

Watson, J., Woodruff, D., and Hart, W. (2012). Pysp: Modeling and solving stochastic programs in python. *Mathematical Programming Computation*, 4(2):109–149.

Word, D., Abbott III, G., Cummings, D., and Laird, C. (2010). Estimating seasonal drivers in childhood infectious diseases with continuous time and discrete-time models. In Basar, T., editor, *Proceedings of ACC 2010*, pages 5137–5142, Baltimore. AACC.

Word, D., Cummings, D., Burke, D., Iamsirithaworn, S., and Laird, C. (2012). A Nonlinear Programming Approach for Estimation of Transmission Parameters in Childhood Infectious Disease Using a Continuous Time Model. *Journal of The Royal Society Interface*, 9(73):1983–1997.

Xia, Y., Bjornstad, O., and Grenfell, B. (2004). Measles metapopulation dynamics: A gravity model for epidemiological coupling and dynamics. *The American Naturalist*, 164(2):267–281.

Yorke, J. and London, W. (1973). Recurrent outbreaks of measles, chickenpox, and mumps. *American Journal of Epidemiology*, 98(6):469–482.

Zavala, V. (2008). *Computational Strategies for the Optimal Operation of Large-Scale Chemical Processes*. PhD thesis, Carnegie Mellon University, Pittsburgh.

Zavala, V. and Biegler, L. (2006). Large-scale parameter estimation in low-density polyethylene tubular reactors. *Industrial & Engineering Chemistry Research*, 45(23):7867–7881.

Zavala, V., Laird, C., and Biegler, L. (2008). Interior-point decomposition approaches for parallel solution of large-scale nonlinear parameter estimation problems. *Chemical Engineering Science*, 63(19):4834–4845.

Zhu, Y. and Laird, C. (2008). A parallel algorithm for structured nonlinear programming. In Malone, M., Trainham, J., and Carnahan, B., editors, *Proceeding of 5th International Conference on Foundations of Computer-Aided Process Operation, FOCAPO*, pages 345–348. CACHE, American Institute of Chemical Engineers, New York.

Zhu, Y., Legg, S., and Laird, C. (2010). Optimal design of cryogenic air separation columns under uncertainty. *Computers & Chemical Engineering*, 34(9):1377–1384.

Zhu, Y., Legg, S., and Laird, C. (2011a). A multiperiod nonlinear programming approach for operation of air separation plants with variable power pricing. *AIChE Journal*, 57(9):2421–2430.

Zhu, Y., Legg, S., and Laird, C. (2011b). Optimal operation of cryogenic air separation systems with demand uncertainty and contractual obligations. *Chemical Engineering Science*, 66(5):953–963.

Zhu, Y., Word, D., Siirola, J., and Laird, C. (2009). Exploiting modern computing architectures for efficient large-scale nonlinear programming. *Computer Aided Chemical Engineering*, 27:783–788.

APPENDIX A

DISCRETE-TIME FORMULATION IN AMPL

This is an AMPL model for the deterministic, discrete-time formulation described in Section 4. Note that additional parameters are defined in the run file described in Appendix F.

```
set S_SI ordered;
set S_SI_PER_YEAR ordered;
param P_POP{S_SI};
param P_BIRTHS{S_SI};
param P_REP_CASES{S_SI}, default 1.0;
param P_DATA_WTS{S_SI}, default 1.0;
param P_YEARS{S_SI};
param P_NSIPY = card(S_SI_PER_YEAR);
param P_OBJWT_lnC > 0;
param P_BETA_INIT > 0;
param P_SUSC_INIT_PERC > 0;
param P_YEAR_DATA_START > 0;
param P_YEAR_DATA_END > 0;
param P_YEAR_START >= P_YEAR_DATA_START;
param P_YEAR_END > P_YEAR_START, <= P_YEAR_DATA_END;
set S_SI_L ordered =
    ( ((P_YEAR_START - P_YEAR_DATA_START)*card(S_SI_PER_YEAR) + 1)..
    ((P_YEAR_END - P_YEAR_DATA_START + 1)*card(S_SI_PER_YEAR)) by 1);
param P_GLOBAL_GAMMA > 0, <= 1.1;
```

```
var I{S_SI_L} >= 0, <= P_POP[ first(S_SI_L) ];

var S{S_SI_L} >= 0, <= P_POP[ first(S_SI_L) ];

var ln_I{S_SI_L} <= log(P_POP[ first(S_SI_L) ]);

var ln_S{S_SI_L} <= log(P_POP[ first(S_SI_L) ]);

var S_bar >= 0, <= P_POP[ first(S_SI_L) ],

    := P_SUSC_INIT_PERC*P_POP[ first(S_SI_L) ];

var ln_beta{S_SI_PER_YEAR} := log(P_BETA_INIT), >= log(0.5), <= log(70);

var beta{S_SI_PER_YEAR} := P_BETA_INIT, >= 0.5, <= 70;

var beta_bar := P_BETA_INIT, >= 0.5, <= 70;

var alpha := 1, >= 0.75, <= 1.7;

var gamma{S_SI_L} := P_GLOBAL_GAMMA, >=0, <= 1.1;

var gamma_inc := 0;

var ln_gamma{S_SI_L} := log(P_GLOBAL_GAMMA), <= log(1);

var eps_ln_C{S_SI_L} := 0;


minimize obj: 1/2*sum{i in S_SI_L}( P_OBJWT_lnC * eps_ln_C[i]^2 );

s.t.

   DisDynamics{i in S_SI_L diff { first(S_SI_L) } }:

      ln_I[i] = (ln_beta[(((i-1)-1) mod P_NSIPY) + 1]) - log(P_POP[i-1])

                 + alpha*ln_I[i-1] + ln_S[i-1];


   SusDynamics{i in S_SI_L diff { first(S_SI_L) } }:

      S[i] = S[i-1] + P_BIRTHS[i-1] - I[i];


   ln_Meas{i in S_SI_L: P_DATA_WTS[i] > 0}:

      log( max(P_REP_CASES[i],1e-2)) = ln_gamma[i] + ln_I[i] + eps_ln_C[i];
```

```
I_exp{i in S_SI_L}:
   I[i] = exp(ln_I[i]);

S_exp{i in S_SI_L}:
   S[i] = exp(ln_S[i]);

Gamma_Form{i in S_SI_L diff { first(S_SI_L) } }:
   gamma[i] = gamma[i-1]+gamma_inc;

Gamma_exp{i in S_SI_L}:
   gamma[i] = exp(ln_gamma[i]);

beta_exp{i in S_SI_PER_YEAR}:
   beta[i] = exp(ln_beta[i]);

beta_bar_con:
   beta_bar = sum{i in S_SI_PER_YEAR}(beta[i])/card(S_SI_PER_YEAR);

S_bar_con:
   S_bar = sum{i in S_SI_L}(S[i])/card(S_SI_L);
```

APPENDIX B

CONTINUOUS-TIME FORMULATION IN AMPL

This is an AMPL model for the continuous-time formulation using Gauss-Lobatto collocation that is described in Section 5.

```
param a1 := (1/686);

param a2 := (39*sqrt(21)+231);

param a3 := (224);

param a4 := (-39*sqrt(21)+231);

param a5 := (3*sqrt(21)+21);

param a6 := (-16*sqrt(21));

param a7 := (3*sqrt(21)-21);

param a8 := (-3*sqrt(21)+21);

param a9 := (16*sqrt(21));

param a10 := (-3*sqrt(21)-21);

param c1 := (1/360);

param c2 := (32*sqrt(21)+180);

param c3 := (-64*sqrt(21));

param c4 := (32*sqrt(21)-180);

param c5 := (9+sqrt(21));

param c6 := (98);

param c7 := (64);

param c8 := (9-sqrt(21));

param c9 := (-32*sqrt(21)+180);

param c10 := (64*sqrt(21));
param c11 := (-32*sqrt(21)-180);

param P_STEP            >  0;

param P_RIPY            >= 1;

param P_TRI             >= 1;

param P_FEPR            >= 1;

param P_FEPY            >= 1;

param P_TFE             >= 1;

param P_RHO             >  0;

param P_GAMMA           >  0;

param P_Sbar_init       >  0;

param P_I_init          >  0;

param P_YEAR_START      >  0;

param P_YEAR_END        >  P_YEAR_START;

param P_LIFE_EXP        >  0;

param P_YEARS           >= 1;

param P_NUM_BETA        >= 1;

param P_CP              >= 1;


set S_RIPY ordered      := 1..P_RIPY;

set S_FEPR ordered      := 1..P_FEPR;

set S_TRI ordered       := 1..P_TRI;

set S_TFE circular      := 1..P_TRI*P_FEPR;

set S_FEPY ordered      := 1..P_RIPY*P_FEPR;

set S_BETA ordered      := 1..P_NUM_BETA;

set S_CP ordered        := 1..P_CP;
```

```
param P_PHI_OBJ_WT              >= 0;

param P_SUS_OBJ_WT              >= 0;

param P_INF_OBJ_WT              >= 0;

param P_Sbar_INIT_OBJ_WT        >= 0;

param P_I_INIT_OBJ_WT           >= 0;

param P_DATA_WTS{S_TRI}         >= 0;

param P_REP{S_TRI}      <= 1.2, >= 0;

param P_PHI{S_TRI}              >= 0;

param P_POP{S_TFE}              >= 0;

param P_BIRTHS{S_TFE}           >= 0;

param P_BETA{S_TFE union {P_TRI*P_FEPR+1}} >= 0;

param P_REP_CASES{S_TRI};


var phi{S_TFE, S_CP}        >= 0;

var S{S_TFE, S_CP}          <= P_POP[first(S_TFE)],  >= 0;

var I{S_TFE, S_CP}          <= P_POP[first(S_TFE)],  >= 0;

var beta{S_BETA}            <= 70 >= 0.05;

var dbeta{S_BETA};

var betapos{S_BETA}         >= 0;

var betaneg{S_BETA}         >= 0;

var eps_phi{S_TRI};

var eps_I{S_TFE}            >= 1e-8  ;

var S_bar := 1e5            >= 0;

var beta_bar := 10          >= 0;

var psi{S_TFE, S_CP} := 1   >=0;

var Idot{i in S_TFE, j in S_CP};
var Sdot{i in S_TFE, j in S_CP};
var phidot{i in S_TFE, j in S_CP};

minimize obj: 1/2*(
+ P_PHI_OBJ_WT*sum{i in S_TRI}( ( P_DATA_WTS[i]*eps_phi[i]^2 ) )
+ P_INF_OBJ_WT*sum{i in S_TFE}(( ( log(eps_I[i])^2 ) ) )  );
s.t.
    I_interiorpoint_1{i in S_TFE}:
        I[i,2] = a1*( a2*I[i,1] + a3*I[i,3] + a4*I[i,5]
        + P_STEP*( a5*Idot[i,1] + a6*Idot[i,3] +a7*Idot[i,5] ) );
    I_interiorpoint_2{i in S_TFE}:
        I[i,4] = a1*( a4*I[i,1] + a3*I[i,3] + a2*I[i,5]
        + P_STEP*( a8*Idot[i,1] + a9*Idot[i,3] + a10*Idot[i,5] ) );
    I_bound{i in S_TFE diff{last(S_TFE)} }:
        I[i,5] = I[i+1,1];


    I_sys_const_1{i in S_TFE}:
        0 = c1*( c2*I[i,1] + c3*I[i,3] + c4*I[i,5]
            + P_STEP*( c5*Idot[i,1] + c6*Idot[i,2]
            + c7*Idot[i,3] + c8*Idot[i,5] ) );
    I_sys_const_2{i in S_TFE}:
        0 = c1*( c9*I[i,1] + c10*I[i,3] + c11*I[i,5]
            + P_STEP*( c8*Idot[i,1] + c6*Idot[i,4]
            + c7*Idot[i,3] + c5*Idot[i,5] ) );


    S_interiorpoint_1{i in S_TFE}:
```

```
    S[i,2] = a1*( a2*S[i,1] + a3*S[i,3] + a4*S[i,5]
    + P_STEP*( a5*Sdot[i,1] + a6*Sdot[i,3] +a7*Sdot[i,5] ) );
S_interiorpoint_2{i in S_TFE}:
    S[i,4] = a1*( a4*S[i,1] + a3*S[i,3] + a2*S[i,5]
    + P_STEP*( a8*Sdot[i,1] + a9*Sdot[i,3] + a10*Sdot[i,5] ) );
S_bound{i in S_TFE diff{last(S_TFE)} }:
    S[i,5] = S[i+1,1];


S_sys_const_1{i in S_TFE}:
    0 = c1*( c2*S[i,1] + c3*S[i,3] + c4*S[i,5]
        + P_STEP*( c5*Sdot[i,1] + c6*Sdot[i,2]
        + c7*Sdot[i,3] + c8*Sdot[i,5] ) );
S_sys_const_2{i in S_TFE}:
    0 = c1*( c9*S[i,1] + c10*S[i,3] + c11*S[i,5]
        + P_STEP*( c8*Sdot[i,1] + c6*Sdot[i,4]
        + c7*Sdot[i,3] + c5*Sdot[i,5] ) );


phi_interiorpoint_1{i in S_TFE}:
    phi[i,2] = a1*( a2*phi[i,1] + a3*phi[i,3] + a4*phi[i,5]
    + P_STEP*( a5*phidot[i,1] + a6*phidot[i,3] +a7*phidot[i,5] ) );
phi_interiorpoint_2{i in S_TFE}:
    phi[i,4] = a1*( a4*phi[i,1] + a3*phi[i,3] + a2*phi[i,5]
    + P_STEP*( a8*phidot[i,1] + a9*phidot[i,3] + a10*phidot[i,5] ) );
phi_bound{i in S_TFE diff{first(S_TFE)} }:
    phi[i-1,5] = phi[i,1];
phi_initial:
    phi[first(S_TFE),first(S_CP)] = 0;


phi_sys_const_1{i in S_TFE}:
    0 = c1*( c2*phi[i,1] + c3*phi[i,3] + c4*phi[i,5]
        + P_STEP*( c5*phidot[i,1] + c6*phidot[i,2]
        + c7*phidot[i,3] + c8*phidot[i,5] ) );
phi_sys_const_2{i in S_TFE}:
    0 = c1*( c9*phi[i,1] + c10*phi[i,3] + c11*phi[i,5]
        + P_STEP*( c8*phidot[i,1] + c6*phidot[i,4]
        + c7*phidot[i,3] + c5*phidot[i,5] ) );



rep_cases{i in S_TRI diff{first(S_TRI)}: P_DATA_WTS[i] > 0}:
    P_REP_CASES[i] = P_REP[i]*(phi[(i)*P_FEPR, last(S_CP)]
    - phi[(i-1)*P_FEPR, last(S_CP)]) + eps_phi[i];


rep_cases_first{if P_DATA_WTS[first(S_TFE)] > 0 }:
    P_REP_CASES[first(S_TRI)] = P_REP[first(S_TRI)]*
    (phi[(first(S_TRI))*P_FEPR, last(S_CP)]
    - phi[(first(S_TRI)), first(S_CP)]) + eps_phi[first(S_TRI)];


sbar_const:
    S_bar = (sum{i in S_TFE}( sum{j in S_CP}( S[i,j] ) ))
    /(card(S_TFE)*card(S_CP));


beta_bar_const:
    beta_bar = ( sum{i in S_BETA}(beta[i]) )/card(S_BETA);


Idot_diff_eq{i in S_TFE, j in S_CP}:
    Idot[i,j] = psi[i,j] - P_GAMMA*I[i,j];


Sdot_diff_eq{i in S_TFE, j in S_CP}:
    Sdot[i,j] = -psi[i,j] + P_BIRTHS[i];


phidot_diff_eq{i in S_TFE, j in S_CP}:
    phidot[i,j] = psi[i,j];


log_psi_const{i in S_TFE, j in S_CP}:
    psi[i,j] = beta[ P_BETA[i] ]*I[i,j]*S[i,j]/P_POP[i]*eps_I[i];
```

APPENDIX C

PROGRESSIVE HEDGING MODEL FORMULATION

This is a Pyomo model for the continuous-time formulation using Radau collocation that is used in conjunction with PySP for multi-city estimations as described in Section 6.

```
from coopr.pyomo import *


years = 20
beta_py = 26
fepr = 1
fepy = beta_py*fepr
fe = fepy * years
step = 365.0/fepy
model = AbstractModel()


model.P_GAMMA = Param(default=1.0/14.0)
model.P_NUM_BETA = Param(default=beta_py)
model.P_FEPY = Param(default=fepy)
model.P_FE = Param(default=fe)
model.P_CP = Param(default=3)
model.P_STEP = Param(default=step)
model.P_TRI = Param(default=beta_py*years)
model.P_FEPR = Param(default=fepr)
model.S_BETA = RangeSet(1,value(model.P_NUM_BETA))
model.S_FE = RangeSet(1,value(model.P_FE))
model.S_CP = RangeSet(1,value(model.P_CP))
model.S_TRI = RangeSet(1,value(model.P_TRI))

model.P_BETA_NDX = Param(model.S_FE)
model.P_POP = Param(model.S_FE, default=1.0e6)
model.P_REP_FRAC = Param(model.S_TRI, default=0.4)
model.P_REP_CASES = Param(model.S_TRI, default=10.0)
model.P_BIRTHS = Param(model.S_FE, default=100.0)
model.P_DATA_WTS = Param(model.S_TRI, default=1.0)
model.a = Param(model.S_CP, model.S_CP)
model.P_ALL_CASES = Param(model.S_TRI, default = 10.0)

def _initialize_weights(model):
    return ((3249440.0/value(model.P_POP[1]))**2)
model.I_OBJ_WT = Param(initialize=_initialize_weights)
model.PHI_OBJ_WT = Param(default=1.0)

model.init_S_bar = Param(default=1.0e5)
model.init_beta_bar = Param(default=1.0)
model.init_I_init = Param(default=10.0)
model.init_S_init = Param(default=1000.0)
model.init_beta = Param(model.S_BETA, default=1.0)
model.init_beta_pos = Param(model.S_BETA, default=0.0)
model.init_beta_neg = Param(model.S_BETA, default=0.0)
model.init_eps_I = Param(model.S_FE, default=0.0)

model.init_eps_phi = Param(model.S_TRI, default=0.0)

model.init_S = Param(model.S_FE, model.S_CP, default=100.0)

model.init_I = Param(model.S_FE, model.S_CP, default=10.0)

model.init_phi = Param(model.S_FE, model.S_CP, default=0.0)

model.init_Sdot = Param(model.S_FE, model.S_CP, default=1.0)

model.init_Idot = Param(model.S_FE, model.S_CP, default=1.0)

model.init_phidot = Param(model.S_FE, model.S_CP, default=1.0)

model.init_beta_patt = Param(model.S_BETA, default = 0.0)

model.init_beta_int = Param(default = 1.0)

def _init_S_bar(model):

    return value(model.init_S_bar)

def _init_beta_bar(model):

    return value(model.init_beta_bar)

def _init_I_init(model):

    return value(model.init_I_init)

def _init_S_init(model):

    return value(model.init_S_init)

def _init_beta(model,i):

    return value(model.init_beta[i])

def _init_beta_pos(model,i):

    return value(model.init_beta_pos[i])

def _init_beta_neg(model,i):

    return value(model.init_beta_neg[i])

def _init_eps_I(model,i):

    return value(model.init_eps_I[i])
```

```
def _init_eps_phi(model,i):
    return value(model.init_eps_phi[i])
def _init_S(model,i,j):
    return value(model.init_S[i,j])
def _init_I(model,i,j):
    return value(model.init_I[i,j])
def _init_phi(model,i,j):
    return value(model.init_phi[i,j])
def _init_Sdot(model,i,j):
    return value(model.init_Sdot[i,j])
def _init_Idot(model,i,j):
    return value(model.init_Idot[i,j])
def _init_phidot(model,i,j):
    return value(model.init_phidot[i,j])
def _init_beta_patt(model,i):
    return value(model.init_beta_patt[i])
def _init_beta_int(model):
    return value(model.init_beta_int)
def _people_bounds(model,i,j):
    return (0.0, value(model.P_POP[1]))
def _init_people_bounds(model):
    return (0.0, value(model.P_POP[1]))

model.S_bar = Var(initialize=_init_S_bar, bounds=(0,None))
model.beta_bar = Var(initialize=_init_beta_bar, bounds=(0.05,5))
model.I_init = Var(initialize=_init_I_init, bounds=_init_people_bounds)
model.S_init = Var(initialize=_init_S_init, bounds=_init_people_bounds)
model.phi_init = Param(default=0.0)
model.beta_c = Var(initialize = 1.0)
model.beta = Var(model.S_BETA, initialize=_init_beta, bounds=(0.01,5))
model.beta_pos = Var(model.S_BETA, \
              initialize=_init_beta_pos, bounds=(0,None))
model.beta_neg = Var(model.S_BETA, \
              initialize=_init_beta_neg, bounds=(0,None))
model.beta_patt = Var(model.S_BETA, \
              initialize=_init_beta_patt, bounds=(-5,5))
model.beta_int = Var(initialize = _init_beta_int, bounds=(0.01,5.0))
model.alpha = Var(initialize = 0.05, bounds=(-1.0,1.0))
model.eps_I = Var(model.S_FE, initialize=_init_eps_I)
model.eps_phi = Var(model.S_TRI, initialize=_init_eps_phi)
model.S = Var(model.S_FE, model.S_CP, \
              initialize=_init_S, bounds=_people_bounds)
model.I = Var(model.S_FE, model.S_CP, \
              initialize=_init_I, bounds=_people_bounds)
model.phi = Var(model.S_FE, model.S_CP, \
              initialize=_init_phi, bounds=(0,None))
model.Sdot = Var(model.S_FE, model.S_CP, initialize=_init_Sdot)
model.Idot = Var(model.S_FE, model.S_CP, initialize=_init_Idot)
model.phidot = Var(model.S_FE, model.S_CP, \
              initialize=_init_phidot, bounds=(-10,None))
model.FirstStageObj = Var(bounds=(0.0,0.0))
model.SecondStageObj = Var()
```

```
def second_stage_obj_rule(model):
    return (model.SecondStageObj, value(model.I_OBJ_WT) \
    *sum(model.eps_I[i]**2 for i in model.S_FE) \
    + value(model.PHI_OBJ_WT)*sum(value(model.P_DATA_WTS[i]) \
    *model.eps_phi[i]**2 for i in model.S_TRI))
model.compute_second_stage_obj = Constraint(rule=second_stage_obj_rule)

def _obj_rule(model):
    return (value(model.I_OBJ_WT)*sum(model.eps_I[i]**2 \
    for i in model.S_FE) + value(model.PHI_OBJ_WT) \
    *sum(value(model.P_DATA_WTS[i])*model.eps_phi[i]**2 \
    for i in model.S_TRI))
model.obj = Objective(rule=_obj_rule)

def _reported_cases(model,i):
    if i == 1:
        if value(model.P_DATA_WTS[i]) > 0.1:
            return (value(model.P_REP_CASES[i]), \
            value(model.P_REP_FRAC[i]) \
            *( model.phi[i*value(model.P_FEPR),3] - model.phi_init ) \
            + model.eps_phi[i])
        else:
            return Constraint.Skip
    else:
        if value(model.P_DATA_WTS[i]) > 0.1:
            return (value(model.P_REP_CASES[i]), \
            value(model.P_REP_FRAC[i]) \
            *( model.phi[i*value(model.P_FEPR),3] \
            - model.phi[(i-1)*value(model.P_FEPR),3] ) \
            + model.eps_phi[i])
        else:
            return Constraint.Skip
model.con_reported_cases = Constraint(model.S_TRI, rule=_reported_cases)


def _beta_bar(model):
    return (model.beta_bar, sum(model.beta[i] for i in model.S_BETA) \
    /len(model.S_BETA))
model.con_beta_bar = Constraint(rule=_beta_bar)


def _S_bar(model):
    return (model.S_bar, sum(model.S[i,j] for i in model.S_FE \
    for j in model.S_CP)/(len(model.S_FE)*len(model.S_CP)))
model.con_S_bar = Constraint(rule=_S_bar)


def _phidot(model,i,j):
    return (model.phidot[i,j], model.eps_I[i] \
    +model.beta[value(model.P_BETA_NDX[i])] \
    *model.I[i,j]*model.S[i,j]/value(model.P_POP[i]))
model.con_psi = Constraint(model.S_FE, model.S_CP, rule=_phidot)


def _Idot(model,i,j):
    return (model.Idot[i,j], model.phidot[i,j] \
    - model.P_GAMMA*model.I[i,j])
model.con_Idot = Constraint(model.S_FE, model.S_CP, rule=_Idot)

def _Sdot(model,i,j):
    return (model.Sdot[i,j], -model.phidot[i,j] + model.P_BIRTHS[i])
model.con_Sdot = Constraint(model.S_FE, model.S_CP, rule=_Sdot)

def _I_colloc(model, i, j):
    if i > 1:
        return (model.I[i,j], model.I[i-1,value(model.P_CP)] \
        + value(model.P_STEP) \
        *sum(value(model.a[k,j])*model.Idot[i,k] for k in model.S_CP))
    else:
        return (model.I[i,j], model.I_init + value(model.P_STEP) \
        *sum(value(model.a[k,j])*model.Idot[i,k] for k in model.S_CP))
model.con_I_colloc = Constraint(model.S_FE, model.S_CP, rule=_I_colloc)

def _S_colloc(model, i, j):
    if i > 1:
        return (model.S[i,j], model.S[i-1,value(model.P_CP)] \
        + value(model.P_STEP) \
        *sum(value(model.a[k,j])*model.Sdot[i,k] for k in model.S_CP))
    else:
        return (model.S[i,j], model.S_init + value(model.P_STEP) \
        *sum(value(model.a[k,j])*model.Sdot[i,k] for k in model.S_CP))
```

```
model.con_S_colloc = Constraint(model.S_FE, model.S_CP, rule=_S_colloc)


def _phi_colloc(model, i, j):
    if i == 1:
        return (model.phi[i,j], value(model.phi_init) \
        + value(model.P_STEP) \
        *sum(value(model.a[k,j])*model.phidot[i,k] for k in model.S_CP))
    else:
        return (model.phi[i,j], model.phi[i-1,value(model.P_CP)] \
        + value(model.P_STEP) \
        *sum(value(model.a[k,j])*model.phidot[i,k] for k in model.S_CP))
model.con_phi_colloc = \
              Constraint(model.S_FE, model.S_CP, rule=_phi_colloc)


def _scaled_beta(model, i):
    return (model.beta[i], model.beta_c * model.beta_patt[i])
model.con_city_varying_beta = Constraint(model.S_BETA, rule=_scaled_beta)


def _mean_patt(model):
    return (1.0, summation(model.beta_patt)/len(model.S_BETA))
model.con_mean_patt = Constraint(rule=_mean_patt)


def _beta_c(model):
    return (0.75, model.beta_c, 1.5)
model.con_beta_c = Constraint(rule=_beta_c)
```

APPENDIX D

PROGRESSIVE HEDGING SCENARIO TREE

This is a portion of the scenario tree used by PySP to define the multi-city estimation problem described in Section 6. This scenario tree is used with the Pyomo model described in Appendix C. The complete scenario tree is not given to conserve space.

```
set Stages := FirstStage SecondStage;


set Nodes := RootNode
             BathNode
             BirkenheadNode
             BirminghamNode
             BlackburnNode
             BlackpoolNode
               ⋮
             WalsallNode
             WarringtonNode
             WiganNode
             WolverhamptonNode
             YorkNode;


param NodeStage := RootNode   FirstStage
                   BathNode   SecondStage
```

```
                    BirkenheadNode   SecondStage
                    BirminghamNode   SecondStage
                    BlackburnNode   SecondStage
                    BlackpoolNode   SecondStage
                        ⋮
                    WalsallNode   SecondStage
                    WarringtonNode   SecondStage
                    WiganNode   SecondStage
                    WolverhamptonNode   SecondStage
                    YorkNode   SecondStage;

set Children[RootNode] :=
BathNode
BirkenheadNode
BirminghamNode
BlackburnNode
BlackpoolNode
    ⋮
WalsallNode
WarringtonNode
WiganNode
WolverhamptonNode
YorkNode;
```

```
param ConditionalProbability := RootNode  1.0
                          BathNode         0.016666666
                          BirkenheadNode       0.016666666
                          BirminghamNode       0.016666666
                          BlackburnNode  0.016666666
                          BlackpoolNode  0.016666666
                            ⋮
                          WalsallNode         0.016666666
                          WarringtonNode       0.016666666
                          WiganNode          0.016666666
                          WolverhamptonNode     0.016666666
                          YorkNode         0.016666666;

set Scenarios :=
Bath
Birkenhead
Birmingham
Blackburn
Blackpool
    ⋮
Walsall
Warrington
Wigan
Wolverhampton
```

```
York;

param ScenarioLeafNode :=
Bath   BathNode
Birkenhead   BirkenheadNode
Birmingham   BirminghamNode
Blackburn   BlackburnNode
Blackpool   BlackpoolNode
    ⋮
Walsall   WalsallNode
Warrington   WarringtonNode
Wigan   WiganNode
Wolverhampton   WolverhamptonNode
York   YorkNode;

set StageVariables[FirstStage] := beta[*] ;

param StageCostVariable :=
FirstStage FirstStageObj
SecondStage SecondStageObj;

param ScenarioBasedData := True;
```

APPENDIX E

NEW YORK CITY DATA FILE

This is a portion of the New York City data used in the discrete-time estimations. The complete data was not included due to space.

```
set S_SI = 1 2 3 4 5 ... 1140 1141 1142 1143 1144 ;


param P_POP :=

1 6.71164e+06

2 6.71655e+06

3 6.72147e+06

4 6.72639e+06

5 6.7313e+06

⋮

1140 7.74345e+06

1141 7.74029e+06

1142 7.73712e+06

1143 7.73395e+06

1144 7.73078e+06;


set S_SI_PER_YEAR = 1 2 3 4 5 ... 22 23 24 25 26 ;


param P_YEARS :=

1 1928.038462
```

```
2 1928.076923

3 1928.115385

4 1928.153846

5 1928.192308

⋮

1140 1971.846154

1141 1971.884615

1142 1971.923077

1143 1971.961538

1144 1972.000000;


param P_REP_CASES :=

1 190.574987

2 352.064651

3 481.877080

4 765.584558

5 1404.929095

⋮

1140 7.952320

1141 5.059851

1142 5.986192

1143 8.833161

1144 10.951421;
```

```
param P_DATA_WTS :=
1 1.000000
2 1.000000
3 1.000000
4 1.000000
5 1.000000
⋮
1140 1.000000
1141 1.000000
1142 1.000000
1143 1.000000
1144 1.000000;

param P_BIRTHS :=
1 4860.461538
2 4860.461538
3 4860.461538
4 4860.461538
5 4860.461538
⋮
1140 5070.384615
1141 5070.384615
1142 5070.384615
1143 5070.384615
1144 5070.384615;
```

APPENDIX F

AMPL RUN FILE FOR DISCRETE-TIME ESTIMATION

This is an AMPL run file to run the deterministic, discrete-time estimation described in Section 4. Note that the model included is described in Appendix A, and the data included is shown in Appendix E.

```
param data_name symbolic, = 'nyc_measles_1928_1972_26si';
param output_name symbolic, = 'nyc_measles_det';
model ../../ampl_models/seasonal_beta_stoch.mod;
let P_OBJWT_lnC := 1.0;
let P_BETA_INIT := 15;
let P_SUSC_INIT_PERC := 0.06;
let P_YEAR_DATA_START := 1928;
let P_YEAR_DATA_END := 1971;
let P_YEAR_START := 1944;
let P_YEAR_END := 1963;
let P_GLOBAL_GAMMA := 1/(sum{j in S_SI_L}(P_BIRTHS[j])
           /sum{k in S_SI_L}(P_REP_CASES[k]));
display sum{j in S_SI_L}(P_BIRTHS[j]);
display sum{j in S_SI_L}(P_REP_CASES[j]);
display P_GLOBAL_GAMMA;
param mean_rep_i := sum{i in S_SI_L}(P_REP_CASES[i])/card(S_SI_L);
param prep;
for {i in S_SI_L}
{
   let P_REP_CASES[i] := max(P_REP_CASES[i],1e-2);
   let prep := P_REP_CASES[i];
        if (P_DATA_WTS[i] == 0) then
        {
            let prep := mean_rep_i;
        }
   let I[i] := prep/gamma[i];
   let S[i] := P_SUSC_INIT_PERC*P_POP[i];
   let ln_I[i] := log(I[i]);
   let ln_S[i] := log(S[i]);
}

option ipopt_options "halt_on_ampl_error=yes";
option solver ipopt;
solve;

# Set the grid starting points for generating confidence regions
param Siter := 100;
param Biter := 50;
param mean_population := sum{i in S_TFE}(P_POP[i])/(card(S_TFE));
param start_S_bar_perc := 0.1;          #percentage of total population
param end_S_bar_perc := 0.125;          #percentage of total population
param start_beta_bar := 0.55;
param end_beta_bar := 0.75;
param start_S_bar := start_S_bar_perc*mean_population;
```

```
# Set how far the grid spans
param S_bar_change := (end_S_bar_perc - start_S_bar_perc)*
        mean_population/Siter;
param beta_change := (end_beta_bar - start_beta_bar)/Biter;
```

APPENDIX G

LOG-LIKELIHOOD CONFIDENCE REGION CODE

This is the AMPL code used to generate the log-likelihood confidence regions used in Sections 4 and 5. Parameters not defined herein are defined in the run file.

```
param mean_reg_eps_phi;

param std_dev_reg_eps_phi;

param mean_reg_eps_I;

param std_dev_reg_eps_I;

param lstar_reg;

param l1_reg;


let mean_reg_eps_phi := 1/(card(S_TRI))*sum{i in S_TRI}( (eps_phi[i]) );

let std_dev_reg_eps_phi := ( 1/( card(S_TRI) )*

        sum{i in S_TRI}(( (eps_phi[i])^2 )) )^(0.5);

let mean_reg_eps_I := 1/(card(S_TFE)*card(S_CP))*

        sum{i in S_TFE}( sum{j in S_CP}( log(eps_I[i,j]) ));

let std_dev_reg_eps_I := ( 1/( card(S_TFE)*card(S_CP) )*

        sum{i in S_TFE}(sum{j in S_CP}( (log(eps_I[i,j]))^2 )) )^(0.5);

let lstar_reg := 1/(2*std_dev_reg_eps_phi^2)*

        sum{i in S_TRI}(( (eps_phi[i])^2 ) ) + 1/(2*std_dev_reg_eps_I^2)*

        sum{i in S_TFE}(sum{j in S_CP}( (log(eps_I[i,j]))^2 ) );


param P_OPT_OBJ_FUNC;

let P_OPT_OBJ_FUNC := obj;
```

```
param obj_func{1..Biter,1..Siter};

param l1_reg_beta_bar_S_bar{1..Biter,1..Siter};

param log_like_reg_beta_bar_S_bar{1..Biter,1..Siter};

param beta_range_bb_sb{1..Biter};

param S_range_bb_sb{1..Siter};

param opt_beta_bar;

param opt_S_bar;


let opt_beta_bar := beta_bar;

let opt_S_bar := S_bar;

let beta_bar := start_beta_bar;

let S_bar := start_S_bar;


fix S_bar;

fix beta_bar;


for{i in 1..Biter by 2}

{

   for{j in 1..Siter}

   {

      solve;

      let l1_reg_beta_bar_S_bar[i,j] := 1/(2*std_dev_reg_eps_phi^2)*

              sum{k in S_TRI}(( (eps_phi[k])^2 ) )

              + 1/(2*std_dev_reg_eps_I^2)*

              sum{k in S_TFE}(sum{l in S_CP}( (log(eps_I[k,l]))^2 ) );

      let log_like_reg_beta_bar_S_bar[i,j] :=
              2*(l1_reg_beta_bar_S_bar[i,j]-lstar_reg);
      let S_range_bb_sb[j] := S_bar;
      let S_bar := S_bar + S_bar_change;
      let obj_func[i,j] := obj;
   }

   let S_bar := S_bar - S_bar_change;
   let beta_range_bb_sb[i] := beta_bar;
   let beta_bar := beta_bar + beta_change;
   for{j in 1..Siter}
   {
      solve;
      let l1_reg_beta_bar_S_bar[i+1,Siter+1-j] := 1/2*
              (std_dev_reg_eps_phi^2)*sum{k in S_TRI}(((eps_phi[k])^2))+
              1/(2*std_dev_reg_eps_I^2)*
              sum{k in S_TFE}(sum{l in S_CP}( (log(eps_I[k,l]))^2 ) );
      let log_like_reg_beta_bar_S_bar[i+1,Siter+1-j] :=
              2*(l1_reg_beta_bar_S_bar[i+1,Siter+1-j]-lstar_reg);
      let S_range_bb_sb[Siter+1-j] := S_bar;
      let S_bar := S_bar - S_bar_change;
      let obj_func[i+1,Siter+1-j] := obj;
   }

   let beta_range_bb_sb[i+1] := beta_bar;
   let beta_bar := beta_bar + beta_change;
   let S_bar := start_S_bar;
```

```
}

let S_bar := opt_S_bar;

let beta_bar := opt_beta_bar;

unfix beta_bar;

unfix S_bar;


solve;
```