Computational Scientific Discovery Using Rate-Based Process Models

Adam Arvay

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy in Computer Science,

The University of Auckland, 2018

Abstract

The work discussed in this thesis aims to improve the ability of humankind to understand complex systems by automating the model building process. These techniques fall within the discipline of computational scientific discovery, which has its roots in the early development of artificial intelligence. The approaches developed in this thesis differ from most existing automated model building techniques because they place an emphasis on discovering and expressing models in terms that human beings can easily understand.

The primary problem addressed by this work is that many existing computational scientific discovery approaches are too computationally intensive. The proposed solution involves developing a new rate-based process modelling framework. It is based around the idea of using linear regression to estimate parameter values instead of gradient descent. The regression-based parameter estimation algorithm is implemented into a system called Regression-guided Process Modeller (RPM) that is tested on both empirical and synthetic data. Results from testing demonstrate that RPM is able to find models more than 83,000 times faster than the existing state-of-the-art system.

This work goes on to describe two major additional improvements to the initial implementation as well as the challenge of creating synthetic data. The first improvement was implemented in the system called APM that allowed an existing known model to be used as the starting point to construct new models that explain unseen data. The second improvement in the system Selection-based Process Modeller (SPM) changed the search technique from a greedy technique to a backwards selection based heuristic. Empirical evidence is provided to demonstrate the effect of the improvements compared to the initial system RPM. Synthetic models and data are necessary to evaluate the discovery stems and their creation requires overcoming many of the challenges that computational scientific discovery is intended to automate.

The new modelling framework, and the subsequent working discovery systems RPM, APM and SPM, provide a proof of concept that rate-based process modelling framework can find models more efficiently that previous approaches. These initial systems provide a starting point for the application of the rate-based process modelling framework as a computational scientific discovery tool to real-world dynamic systems.

Acknowledgements

A very special thanks to Mark Gahegan. This thesis could not have been completed if not for his entirely selfless contributions of time, energy, and even office space. I would also like to express my gratitude for the important contributions of Pat Langley who, along with the many critical technical contributions to the research and writing, provided me with a rare and life changing opportunity to study and live in New Zealand.

Many others played an important part in helping me through the difficult challenge of doing an intense PhD in a new discipline from an island on the far side of the world. Thanks to my mother Lisa for making the enormous effort to take not one but two trips to New Zealand during my time there along with providing a great home for my dog. Thanks to my dad and stepmom Rich and Laurie for providing me a place to live during my various transitions across the planet and all the other support. Thanks to Julie for listening to me while I complained constantly about all the goofy and unexpected things that seemed to be happening in my life all the time. Thanks to Keri for helping to learn the importance of gratitude. My old office mate, now Doctor Tania, always made me feel better about how bad my PhD was going because hers was always going worse (congrats). Thanks to Kas who showed me what New Zealand life outside of the university was all about. I also must recognize my fellow airmen who were there with me at Keesler AFB and whose companionship was greatly appreciated during that stressful period (chomp). Finally, I want to express a general thankfulness to the universe for my good fortune in having so many things that are out of my control work out as well as they have.

Contents

Abstract ii
Acknowledgements iii
Contents iv
List of figures and tables vii
1. Background and introduction 1
1.1. The problem with process models 2
1.2. Aims and significance 3
1.3. Objectives 4
1.4. Outline of the thesis 4
2. Existing relevant computational scientific discovery literature 6
2.1. Data mining and machine learning 8
2.2. System identification 11
2.3. Equation discovery 12
2.4. System dynamics 14
2.5. Qualitative reasoning 15
2.6. Quantitative process models 17
2.7. Concluding remarks 20
3. An improved parameter estimation algorithm 22
3.1. The problem with SC-IPM 22
3.2. The development of a regression based parameter identification algorithm 26
3.2.1. Process based representation that emphasizes regression 27
3.2.2. Key assumptions and the method to identify parameters 28
3.3. Sources of error 29
3.3.1. Derivative estimation error 30
3.3.2. Simulation 31
3.4. Identifiability of processes 33
3.4.1. Experimental results 34
3.5. Noise 38
3.6. Concluding remarks 40
4. Heuristic induction of rate-based process models 42
4.1. Simplifying assumptions for a discovery system 42

4.2. Notation 43
4.3. Regression-guided process modeller 46
4.4. Theoretical claims 49
4.5. Evaluation of the Approach 49
4.5.1. Complexity analysis 50
4.5.2. Empirical behaviour on natural data 54
4.5.3. Induction with irrelevant processes 55
4.5.4. Handling noise and complex models 56
4.5.5. Comparison to SC-IPM 58
4.6. Limitations and next steps 60
5. The creation of synthetic data 62
5.1. The need and the challenge of synthetic data 62
5.2. Selecting components of a synthetic process model 64
5.3. Simulation algorithm 65
5.4. The model building method 70
5.4.1. A predator prey domain example 70
5.5. Concluding remarks 74
6. Heuristic adaptation of rate based process models 75
6.1. Introduction 75
6.2. An approach to process model adaptation 76
6.2.1. Detecting Anomalous Derivatives 76
6.2.2. Revising Equation Parameters 78
6.2.3. Adapting Equation Structure 80
6.3. Experimental Evaluation 83
6.3.1. Basic Functionality 83
6.3.2. Generality of the Approach 84
6.3.3. Computational Benefits 87
6.4. Related Research on Model Revision 89
6.5. Concluding Remarks 90
7. Induction of rate based process models using a selection heuristic 91
7.1. A Fourier analysis based selection algorithm 92
7.2. A backwards elimination selection heuristic 97
7.3. Other features of SPM 98

7.4. Experimental evaluation of the selection algorithm 98
7.5. Scalable induction of differential equations 101
8. Conclusion 104

List of figures and tables

Figure 1 The model structure as reported by SC-IPM when given data of two interacting species in a predator-prey ecosystem.....23
Figure 2 Synthetic data of a two variable predator prey system.....24
Figure 3 Distribution of model scores reported by SC-IPM when asked to perform a single parameter search loop.....25
Figure 4 Equation format of a process model formulated to be solved using regression.....29
Figure 5 A demonstration of the effects of bias due to using a forward estimation 31
Figure 6 Graphical representations of the difference between model trajectories when using the actual known parameter values and the parameter values estimated from the regression based algorithm. Calculated total sum of squares error at time step 0.5 and 0.25 are also shown.....33
Figure 7 The trajectories that result from the best fit parameter values when the wrong equation structure, which uses $P_1 = conc_a^2$, is provided to the algorithm.....34
Figure 8 Trajectories resulting from parameter estimation when a model structure that uses $P_1 = conc_a/conc_n$ is given. The second graph depicts the mathematical equivalency between the original data and the ratio of the variables multiplied by a scalar that accounts for the algorithm finding a very good fit.....36
Figure 9 The trajectories that result from the best fit parameter values when the wrong equation structure, which uses $P_1 = conc_n/conc_a$, is provided to the algorithm.....37
Figure 10 The trajectories that result from the best fit parameter values when the wrong equation structure, which uses $P_1 = conc_a * (1 - 0.0022 * conc_a)$, is provided to the algorithm.....37
Figure 11 The trajectories that result from the best fit parameter values when the wrong equation structure, which substitutes $P_2 = conc_a/conc_n$, is provided to the algorithm.....38
Figure 12 Graphical representation of the model trajectories when five percent noise is added to the data. The table shows the corresponding parameter values from the underlying noise free mode, the best fit parameters returned by the algorithm and the reported r^2 error scores.....39
Figure 13 Graphical representation of the model trajectories when ten percent noise is added to the data. The table shows the corresponding parameter values from the underlying noise free mode, the best fit parameters returned by the algorithm and the reported r^2 error scores.....39
Figure 14 Graphical representation of the model trajectories when twenty percent noise is added to the data. The table shows the corresponding parameter values from the underlying noise free mode, the best fit parameters returned by the algorithm and the reported r^2 error scores.....40
Figure 15. A process diagram for the predator prey ecosystem model in Table 4. Processes are enclosed by rectangles and variables in ellipses. Directed arrows indicate the influence of the process on a variable's derivative.....44
Figure 16 An example generic process specification in the RPM system. These are used to generate process instances when definitions of predator and prey variables are provided by the user 46
Figure 17 (a) An empirical determination of the growth rate of the total number of process instances when the number of generic processes is held constant and the number of variables is increased. (b) A similar analysis when variables are held constant and processes are increased. Power series trend lines are fit to each curve and the trend line equation and r^2 statistic is reported.....52
Figure 18 Observed estimated derivatives and model predicted derivative values for a two variable predator-prey ecosystem.....55

Figure 19 Observed measurements and simulated model trajectories for a two variable predator-prey eco system. 55
Figure 20 Simulated trajectories of a five variable predator prey linear food chain ecosystem..... 56
Figure 21 CPU time required for RPM to induce a model for increasingly complex systems. 58
Figure 22 Performance of RPM vs SC-IPM on a model induction task. 59
Figure 23. A process diagram for the predator prey ecosystem model in Table 1. Processes are enclosed by rectangles and variables in ellipses. Directed arrows indicate the influence of the process on a variable's derivative. 64
Figure 24 An expanded predator prey ecosystem model constructed by expanding a simpler model found using empirical data 65
Figure 25 Matrix and differential equation formats of a process model 65
Figure 26 The initial derivative calculation of the 4th order Runge-Kutta method shares similarities with the Euler method, which estimates the functional value x at h time units into the future using a single derivative estimate. See text for further details. 66
Figure 27 The derivative approximation of the second stage of the RK4 algorithm. See text for more details. 67
Figure 28 The third derivative approximation in the RK4 algorithm 68
Figure 29 the fourth approximation of the RK4 algorithm taken at t=h 69
Figure 30 The final approximation is a weighted average of the four approximation with higher weight being given to the midpoint derived derivative estimates. 70
Figure 31 Trajectories of the model defined by Table 5 where initial values all start at 1.0 72
Figure 32 The model structure is identical to the one that produced the trajectories in Figure 31 but the initial value of the variable x3 was changed to 0.5, as can be seen on the grey line with triangle markers starts at 0.5 where all of the others start at 1.0. 73
Figure 33 trajectories produced using the coefficient matrix from Table 6 74
Figure 34 Workflow of the adaptation method. An existing model along with new data is provided and evaluated to identify anomalous equations. Parameters are revised to fit the new data and failing that, structure is revised. 76
Figure 35 Trajectories for the six-variable predator-prey system from Table 7. 77
Figure 36 Trajectories for the six-variable target predator-prey model in Table 8. 79
Figure 37 Observed vs. predicted derivatives for four variables from the model in Table 7. Acceptable equations have points that fall along the diagonal line, whereas ones that require revision are widely scattered. The reported r^2 scores for dx1 dx2 and dx3 were -3.13, 0.22, and -4.84. 81
Figure 38 Process diagram for the six-variable target predator-prey model in Table 8. 82
Figure 39 Process diagram of the aquatic ecosystem model. The target model includes an exogenous growth process, indicated by dashed outlining, that influences the phyto variable. 85
Figure 40 Trajectories of the target aquatic ecosystem model from Table 9. 87
Figure 41 APM's processing time in milliseconds as a function of the number of variables in the target model when equations for variables require parameter revision and structure revision. The higher third curve in the left chart gives the time needed for RPM to induce the same models from scratch. The right chart shows a zoomed in view of the two revision curves. 88
Figure 42 The wave-like appearance of the process instance trajectories R1 and R2 that combine linearly to generate the trajectory of the derivative dx1. 92

Figure 43 The Fourier transform frequency histogram of the first 9 process instances and the derivative of x1. Lines connect each histogram bin. The copious amounts of information provide a demonstration of the difficulty of extracting useful information from a fourier transform of the data. 93
Figure 44 The Fourier transform of the five process instance trajectories along with the derivative of dx1 which shows more detail 95
Figure 45 A normalized Fourier transform frequency histogram of several process instances along with dx2 96
Figure 46 Observed trajectories for a 20-variable predator prey system. The large number of variables illustrates the necessity of using computational scientific discovery system to understand the dynamics. 99
Figure 47 Trajectories from a chemical system involving six interacting chemicals with equations shown in Table 10 101
Figure 48 Trajectories for a chemical systems with ten variables interacting through 12 reactions. 101
Figure 49 Average time for SPM and RPM to find target equations, in CPU seconds. 102

Table 1. Summary table of approaches related to computational scientific discovery 21
Table 2 Results from initial reliability experiments of the SC-IPM parameter estimation algorithm when given the model structure and asked to find parameters 26
Table 3 A conversion table for names of objects in the SC-IPM model translated to clarify how a process model can be expressed in a format that can be solved using multiple regression. Part (a) shows the name of the parameters as defined by SC-IPM along with the new parameter name as it appears in a regression equation. Part (b) demonstrates how a process model can be expressed in a form that can be solved using regression. Part (c) gives the definition of the process rates so they can be calculated from the observed concentrations. 27
Table 4. A three process model for a predator prey eco system in process notation, matrix notation, and differential equation notation 45
Table 5 The components of a process model with six variables that has the model structure shown in Figure 24 71
Table 6 A different coefficient matrix for the model structure defined in Table 5 that produces the trajectories in Figure 33 73
Table 7 (a) Quantitative process model for a six-variable predator-prey ecosystem and (b) the set of six differential equations into which the model compiles. 77
Table 8 (a) Quantitative process model for a six-variable ecosystem that is the target for revision and (b) the set of differential equations into which the model translates. Differences from the base model in Table 2 are indicated in boldface font and underlined.
_____ 80
Table 9 (a) Quantitative process model for the base aquatic ecosystem, with differences between the base and target model are indicated in boldface text and underlined. (b) The set of differential equations into which the model translates. Equation 3T shows the parameter values used in the target model equation 86
Table 10 Differential equations for a chemical system with six variables that interact through eight distinct reactions. SPM can reconstruct this model from the time series whereas RPM cannot. 100

THE UNIVERSITY OF AUCKLAND
SCHOOL OF GRADUATE STUDIES

Co-Authorship Form

Graduate Centre
The ClockTower – East Wing
22 Princes Street, Auckland
Phone: +64 9 373 7599 ext 81321
Fax: +64 9 373 7610
Email: postgraduate@auckland.ac.nz
www.postgrad.auckland.ac.nz

This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work**. Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work. Computational Scientific Discovery by Heuristic Adaptation of Rate-Based Process Models	
Nature of contribution by PhD candidate	Writing sections 2-6, Algorithm prototyping and implementation, figures, experimental work, development of data sets
Extent of contribution by PhD candidate (%)	90

CO-AUTHORS

Name	Nature of Contribution
Pat Langley	Writing section 1, Algorithm design, editing

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- that the candidate wrote all or the majority of the text.

Name	Signature	Date
Pat Langley	Patrick W Langley	8/3/2017

1. Background and introduction

Many scientific and engineering domains involve working with multi-variate time series data. Reasoning about these types of dynamic systems is a high-level cognitive task that often requires a great deal of knowledge and intelligence. The goal of many scientists is to reveal a novel discovery that increases understanding of a dynamic system. The prevalence and difficulty of working with these kinds systems calls for the continued development of the field of computational scientific discovery. This field seeks to automate some aspects of discovery to leverage computational power to speed up the discovery process.

The need for computational scientific discovery is more critical in the age of big data. Humanity can collect data much faster than they are able to analyse and understand it. Entire fields such as machine learning and data mining have developed in part to address the explosion in available data to try to make sense of it. These approaches largely neglect the importance of deep understanding of the dynamics of the system responsible for creating the data that is collected so voraciously.

Computational scientific discovery is distinguished from mainstream work in data mining and machine learning in two important ways. First, an objective of scientific discovery is to uncover new knowledge and increase the understanding of the system—rather than simply predicting outcomes with high accuracy. This generally requires a causal model that produces accurate predictions on unseen data. The causal explanation of why the system behaves this way and not some other way is of central importance for discovery. Descriptive summaries of data based on statistical regularities make no claims about causal influences between variables thus cannot satisfy the requirement to explain.

The other distinguishing factor is that it produces models that are stated in scientific formalisms related to the domain being modelled. Using a familiar formalism increases the odds that the scientist will find the model acceptable (Pazzani, Mani, & Shankle, 2001). Data mining and machine learning algorithms create models using formalisms that only data scientists understand and make no connection to the systems they are modelling. So called black-box or grey-box models where the internal mechanisms of the model are inscrutable to human beings are not acceptable.

The central pillar of many scientific enterprises is understanding how a dynamic system works. That understanding is often expressed in a mathematical or computational model. Computational scientific discovery has a wide range of applications across many domains because it aims to build models in the language of the research domain. To be useful, the model must serve as:

1. a communication tool,
2. a prediction tool, and
3. an experimental apparatus.

This is true whether the model is constructed manually by a person or automatically with an algorithm. Many disciplines would benefit from computational scientific discovery tools and techniques that aid in the model building process. Some including ecology and hydrology have already had some success (Borrett, Bridewell, Langley, & Arrigo, 2007; W. Bridewell & Langley, 2010).

Research on computational scientific discovery has been progressing since the 1980's. Some of the earliest work on this task involved discovering numeric laws (Langley, Simon, & Bradshaw, 1987). These initial efforts could not handle time series data. Over time this approach developed into equation

discovery (Džeroski & Todorovski, 2008) where differential equation models are automatically constructed from time series data and some background knowledge such as partial models. Still, this technique does not place an emphasis on creating models in terms that connect to the domain being modelled, instead creating a set of polynomial equations that accurately describe rather than explain the dynamics of an observed system.

Theoretical knowledge about the domain supports the creation of explanatory models that account for observations in deeper terms. This knowledge is often stated in terms of processes. A process is a causal relation between variables possibly though theoretical unobserved terms. A dynamic system can be composed from many of these relatively simple processes that interact over time to create the complex observed behaviours. A process-based formalism also helps the discovery because processes form the building blocks used to construct complete models that both predict and explain the observed behaviour.

Process model induction is the activity of using a set of individual candidate processes to construct a complete process model. For example, simple chemical reactions can be represented as individual processes and a chemical reaction network could be built from several interacting processes. Research into process model induction has progressed steadily since its first appearance (Langley, Sanchez, Todorovski, & Dzeroski, 2002; Todorovski, Bridewell, Shiran, & Langley, 2005; Borrett et al., 2007; Bridewell & Langley, 2010). Initial systems were able to accept generic knowledge about processes that could appear in the model, along with time-series data for several observed variables. The algorithm could produce a simulation model that reproduces the observed trajectories, predicts future trajectories, and whose process structure explains the activities occurring within the system using terms familiar to domain scientists. The set of processes compiles into a set of differential equations. Unlike equation discovery, the differential equations are organized into distinct processes, with each piece of an equation belonging to a specific process.

Since its introduction, advances in induction techniques have added the ability to incorporate hierarchical knowledge (Todorovski et al., 2005) handle missing values (W. Bridewell, Langley, Racunas, & Borrett, 2006), incorporate constraints to reduce search (Bridewell & Langley, 2010), and even to learn constraints from sample models (Bridewell & Todorovski, 2007). Process modelling induction approaches has been applied to many domains including ecology, hydrology, and biochemistry. Despite the success of using a process model framework for computational scientific discovery, the current induction algorithms suffer from three important shortcomings, described in the following section.

1.1. The problem with process models

The current approach to inducing process models faces three fundamental challenges. Despite the compositional nature of process models, only complete model structures can be evaluated. Current algorithms construct all possible models up to a certain complexity then begin evaluating each one, resulting in an exhaustive search. An exhaustive search is feasible when there are relatively small numbers of variables and generic processes, resulting in just thousands of alternative model structures. But the number of alternative model structures grows exponentially with the number of variables and generic processes, so this approach will not scale.

A second issue is the evaluation of a model requires fitting the unknown parameters in the model to the data. A standard approach to fitting parameters involves using gradient descent techniques in order to

minimize error between the model's predictions and observed data. Each step of the gradient descent algorithm requires a full simulation of the entire model across the test data. Optimization of a single model requires multiple simulations of that model. In most runs of the induction system SC-IPM (Bridewell & Langley, 2010), 99.99% of the CPU time was spent running simulations for parameter estimation.

The final issue is that often times the parameter space is non-linear so the optimization algorithm can find, and get stuck in, local optima. Techniques to search the parameter space more thoroughly for the global optima require even more computational effort. These computational efforts must be repeated for every candidate model and the computational burden only grows as the models become more complex and contain more parameters. As a final insult, after all of this search effort there is no guarantee that the optimal parameter values will be found.

1.2. Aims and significance

These problems have rendered previous attempts to induce process models, such as SC-IPM, too computationally expensive to effectively handle large data sets with many variables, or large numbers of candidate processes. The aim of this work is to design a new approach that retains the advantages of a process based representation while being more scalable. It is also important that the new system be able to handle all of the same domains that current systems can.

The new approach will centrally involve developing the concept of a rate-based process, a new way of defining processes for use in constructing process models. This formulation of processes should address the scalability concern while still retaining the explanatory power and flexibility of a process based representation. This formulation of processes will address many of the critical judgements that have made of the current body of work by demonstrating that the new framework for inducing process models is able to handle models containing more variables and processes than anything described in the literature relevant to this subject.

This work makes four significant contributions to the important field of computational scientific discovery.

1. The development of regression based parameter estimation algorithm that addresses some of the major issues facing current process based discovery systems.
2. The regression based parameter estimation algorithm is implemented into a process modelling framework that greatly improves the computational efficiency, allowing the new approach to scale to much more complex models containing at least 20 variables.
3. The development of an adaptation algorithm that allows existing models to be adapted to new data, providing further computational benefit and reduced human effort.
4. Finally, a demonstration of a model induction heuristic that greatly improves the complexity of the models that the system can handle.

Several of these contributions have been individually described in research papers that have been peer reviewed and presented and/or published in international conferences and journals for artificial intelligence (Langley and Arvay 2015; Arvay and Langley 2015; Arvay and Langley 2016).

1.3. Objectives

The objective of this thesis is to empirically measure the above contributions to demonstrate that they are significant.

1. Provide detailed documentation of the performance and reliability of the SC-IPM based approach using a simple example, illustrating the need for a different approach.
2. Provide evidence of the feasibility of a regression-based parameter estimation approach that shows the algorithm can estimate parameters under suboptimal conditions and will reject equations with wrong terms by returning a poor fit.
3. Provide evidence that the regression-based process modeller (RPM) system that implements the new framework has equivalent coverage as the previous system SC-IPM by running both systems on equivalent data and comparing the results. In addition, gather measurements of the CPU time taken to produce those results to provide evidence as to whether the RPM system is significantly faster.
4. Demonstrate that the model adaptation capabilities, which alter an existing known model to a new setting, is a natural consequence of the new framework then provide evidence using synthetic data that adapting an existing model to new data offers significant CPU time savings over constructing a model from scratch.
5. Address the poor scaling performance of exhaustive search strategy of the initial approach by implementing a selective induction algorithm and demonstrate that it is faster in terms of CPU time while maintaining equivalent coverage.

1.4. Outline of the thesis

The next chapter provides a review of the relevant literature. Computational scientific discovery ultimately produces models which come in a wide variety of forms and purposes across many different domains. Some of the most directly related modelling approaches include data mining and machine learning, system identification, equation discovery, system dynamics, qualitative reasoning, and quantitative process models. The different features of each modelling approach that make them more or less useful as a computational scientific discovery tool are discussed.

Chapter three provides evidence of the specific problems with SC-IPM that underlie its scaling and reliability issues. Addressing the issues by modifying the existing approach seemed infeasible, so in response, a completely new regression-based parameter estimation approach is developed, and initial performance testing is undertaken. Results from the initial testing show that the regression-based approach is able to accurately identify parameters under a variety of conditions.

Chapter four describes a LISP implementation of the regression-based parameter estimation algorithm into a framework that uses newly developed concept of rate-based processes. The chapter clarifies the difference between previous approaches and the new approach. It also shows how these framework changes make a drastic improvement in how model search is executed, overcoming the primary limitation of previous process based work. Synthetic and empirical test data are used to evaluate the performance and functionality of the RPM system, with an emphasis on performance improvements over previous systems.

Chapter five describes the role of synthetic data in the development of this framework and its various implementations and extensions. The need for synthetic data and the challenge of creating it is

described. Creating synthetic data requires performing many of the difficult and tedious steps that computational scientific discovery is intended to automate. The chapter also provides a graphical description of the 4th order Runge-Kutta simulation algorithm.

Chapter six describes an extension to the RPM system that allows it to include an existing model as input and use it as the starting point to explain new data, keeping the parts of the model that are still valid and replacing or altering the processes that do not fit with the new data. The chapter explains how this model adaptation extension leverages one of the most powerful aspects of the rate-based process model formulation. Two sources of synthetic data are used to demonstrate the performance savings by adapting an existing model over building a model from scratch.

Chapter seven describes an extension to RPM that enables selective induction of rate-based process models by replacing the exhaustive search used by RPM with a backwards selection heuristic. The chapter provides evidence that this new search method improves performance while retaining model accuracy on one empirical and 6 synthetic data sets spanning two distinct scientific domains. The new system also returns multiple candidate models that improves the probability that the system will find a model that fits the data.

The final chapter concludes the thesis by reviewing the objectives and the evidence that shows the objectives were met. The chapter also describes future work related the rate-based process model framework.

The general goal of computational scientific discovery involves automation of the scientific discovery process. There are a variety of scientific disciplines, each utilizing a wide variety of techniques, data sources, and scientific outputs. Computational scientific discovery occupies a narrow slice of artificial intelligence that aims to capture and automate the vast variety that exists as part of the human scientific endeavour. The community has its roots during the original development of the artificial intelligence movement during the 1960's. Since the development of this branch of research, it has not grown as much as some of the other areas of artificial intelligence. Computational scientific discovery shares certain aspects with many different research areas but has specific and unique goals that make it distinct from these other areas. Model building remains one of the central elements of computational scientific discovery, a task that is shared across many different domains both in and out of artificial intelligence. This review focuses on the model building aspect of the scientific discovery process and how these aspects can be applied to the task of computational scientific discovery.

2. Existing relevant computational scientific discovery literature

Computational scientific discovery aims to create computer based systems that can make discoveries that are equivalent to human scientists. This aim falls within the broad scope of artificial intelligence (AI). It continues in the vein of the earliest ambitions of the AI movement, sometimes referred to as "good old fashioned AI" which is to create intelligent systems on par with human beings in terms of generality, flexibility, and resilience (Fahlman, 2012). This is in contrast to the more modern approaches to AI research that focus on creating super-human systems that focus on a very narrow task such as playing chess, or solving algebra problems.

There are two primary motivations for studying computational scientific discovery. The first is to increase the understanding of how humans undertake complex cognitive tasks such as discovery. Early AI research began to unravel some of the complex forms of knowledge and techniques that humans brought to bear when solving various problems. Simon & Newell (1958) speculated that heuristic search techniques could be used to solve ill-structured problems. One aspect of a scientific discovery task that makes it ill-structured is that it lacks definite criterion for testing a proposed solution, nor a mechanized solution for applying the criterion (Simon, 1973). At the time, many believed that ill-structured problems were the realm of human problem solvers and so could never be handled by an AI system.

The second motivation involves developing the means to build AI systems that can replicate human general problem solving abilities. Most approaches to general problem solving involve defining an initial problem state and a goal state along with operators that can transform one state into another state. The solution to the problem is the set of operations that transform the initial state to the goal state. An example of a problem that fits this description well is solving a Rubik's cube. An important insight drawn from that early work was the idea that discovery is a special class of general problem solving that could indeed be solved using heuristic search (Langley, Simon, Bradshaw, & Zytkow, 1987). Under Newell's view, human scientists used mental operators to transform one knowledge state into another along with rules of thumb to choose operators, select among candidate states, and decide what qualifies as an acceptable solution (Džeroski, Langley, & Todorovski, 2007). This insight created the opportunity for existing computational problem solving methods to be brought to bear on discovery. It is this second motivation that is explored more thoroughly in this review by examining various approaches that attempt to automate aspects of discovery.

An AI system that can perform computational scientific discovery has several potential benefits. Computers are able to process data much more quickly than a human scientist. Speed has become all the more important in the age of big data. In addition, an artificial system can access more working memory and be able to handle larger data sets. However, the goal is not to replace human scientists; the discovery system should assist human scientists in order to speed up the discovery process. An important requirement of being a good assistant is that all aspects of the computer systems operation should be comprehensible to its human collaborator, especially the output – the resulting model or findings. In addition to speed and comprehensibility, the AI system must aspire to capture the generality and flexibility of a human scientist so that it can apply discovery techniques to different domains of knowledge and data.

There are several ways that a discovery can manifest, depending on what it is that is being discovered. Some of the most important outcomes in the science process include taxonomies, laws, and theories and each of these arises at first because of some kind of discovery or insight. Taxonomies define or

describe concepts from a domain and relationships among those concepts. One well known taxonomy is the organization of biological organisms into domains, phylum, kingdoms on down to families and species. Taxonomies provide the necessary vocabulary and relationships to form a deeper understanding of the domain. Often they are some of the earliest discoveries about a domain. Laws are ways to summarize relations among objects in the domain. Usually laws do not attempt to answer a why question, they simply specify what happens. For example, the ideal gas law states a relationship between pressure, volume, temperature and molar quantity. The law itself makes no claims as to why it is true, it provides a description of observed behaviour. Laws build on taxonomies by describing a more formal and detailed relationship between domain objects. Theories are statements about structures and processes that arise in an environment. Unlike laws, they make a claim about why the relationship takes the form that it does. Theories are usually stated in terms found in the domain taxonomy and the theory interconnects domain laws in a unified account – such as a system of equations or logic. In many cases it is necessary for the theory to be consistent with other accepted theories in order for it to gain acceptance. For example, the theory of energy conservation is a critical aspect of the understanding of physics, and a theory from any domain that purports to violate energy conservation will likely be met with harsh criticism. A theoretical account provides the framework necessary for an explanation to be formed about the cause of the behaviour of the observed system. In summary, laws describe relationships while theories explain them.

The approach to discovery described in this thesis focuses on creating a mathematical model of some system. A model is a simplified representation of a real phenomenon, object, or process. It is a special case of a law or theory applied to a particular situation. The models of most interest serve both as communication tools and prediction tools. Constructing the model by incorporating knowledge structures from the domain theory allows the model to be understood by, and communicated between, human researchers. Prediction allows the model to be tested for validity as well as aid in decision making about future events. Every model is an approximation of some real system so error should be expected, the amount of error that is acceptable depends on the purpose of the model. Accurate predictions by the model can serve as compelling evidence that the theoretical framework underlying the model provides a correct account of the dynamics of the system.

The general approach to creating a mathematical model using an AI system involves first encoding a theoretical framework that defines a space of alternative models that can account for measured data (observations). Part of the knowledge needed to encode a framework includes defining the important variables in the theory as well as the plausible relationships between the variables, and the form that the relationships take. For example, in some cases correlations between variables serve as a satisfactory relationship. Each model within the space would describe a different set of relationships. Once the space is defined, an exhaustive search of all the possible alternative models would guarantee that a model that fits the data best will be found, if it exists. A computer can search much more quickly than a human, however in most cases there are so many possibilities it would still take an impractical amount of time to exhaustively search the entire space. It is necessary to employ an intelligent strategy that focuses the search effort on models that are more likely to be a solution.

The search strategy, or search heuristic, is an important aspect of computational scientific discovery research, as it is for many other branches of AI, for example machine learning (Michalski, Carbonell, & Mitchell, 1983). The heuristic can be as simple as limiting the maximum number of components of a model, in essence stating that simpler models are preferred over more complex ones. The fundamental

trade-off of using a heuristic is the loss of the guarantee of finding the optimal solution. This trade-off is necessary in order to make it possible that the search will be completed in a reasonable time frame.

One way to satisfy some of the goals of computational scientific discovery is to create an artificial intelligence system that can produce a discovery in the form of a model that can be simulated. The model is an implementation of a theoretical framework that communicates, explains, and predicts the behaviour of the system. Prediction allows the model to be verified against measured data which offers support that the theory underlying the model is correct. In order to automate this process, a theoretical framework has to be encoded and alternative model structures evaluated.

In the following sections I review several areas of artificial intelligence that have tackled different aspects of this broad issue. Data mining and machine learning approaches are some of the most well-known and their methods for automatically constructing models are useful for a variety of applications. System identification is often used in engineering and science domains and shares many of the same goals as computational scientific discovery. Equation discovery is one of the first methods developed specifically for the task of creating models that have some explanatory power. System dynamics provides an example of a unified theoretical approach to building models that can aid in creating explanations. Qualitative process modelling introduces a process based account of systems that greatly improves the explanatory power of their models. Processes are often derived from domain theory which provides a link between the model output and domain theory, thus facilitating explanations. Although this is not the only method of providing explanations, this is the approach taken throughout this thesis. Finally, quantitative process models add much needed predictive power to qualitative process models and set the stage for the work described in the rest of this thesis.

2.1. Data mining and machine learning

Data mining and machine learning are often used together to automate the construction of models from data. Data mining is the process of discovering useful patterns in data and in many cases machine learning techniques are applied automate that process (Witten, Frank, Hall, & Pal, 2016). There is considerable overlap between data mining and machine learning and both fields tend to leverage the speed and processing capabilities of computers to process large amounts of data very quickly. Some of applications include image recognition, fault detection, loan approval, market trend prediction, and more (Dunham, 2003; Gorunescu, 2011).

These approaches generally involve uncovering some hidden structure in the data that helps organize it in a useful way. The data is represented by a model that may be a novel way of organizing the data, which could be considered a discovery. In most cases these approaches do not require searching through a space of alternative model structures. A model structure is selected from the domain of data mining for a data mining task. Most data mining approaches fall under classification, clustering, or association rules (Dunham, 2003).

Classification tasks involve building a model of the data that maps individual data points into broader classes. This is done via supervised learning where classes are learned through training data provided by a human supervisor that provides examples of data that belongs to a given class. The system learns how to classify the training data and then can apply this learned behaviour to unseen data. Each classification algorithm uses different techniques to learn how to partition the attributes in order to assign each instance to the correct class.

Supervised classification techniques fall under four main categories, logic based, neural-network based, probabilistic, and support vector machines (Kotsiantis, Zaharakis, & Pintelas, 2006). Logic based classification approaches are usually applied to data that involves categorical values. These approaches include decision trees (Murthy, 1998; Quinlan, 1986) that sort instances based on feature values and sets of rules which can be translated from a decision tree or created by rule based algorithms (Fürnkranz, 1999). Association rule algorithms are one form of rule-based learning that will be discussed in more detail later in this section. Neural network (Hopfield, 1982; Schmidhuber, 2015) approaches generally operate on continuous data. They are sometimes called universal function approximators because they are able to learn an approximate relationship between the attributes of an object and its group membership (Zhang, 2000). Perceptrons are related systems that are only able to handle linear mappings and often can only produce a binary classification output (Kotsiantis et al., 2006). Probabilistic approaches such as Bayesian networks (Friedman, Geiger, & Goldszmidt, 1997; Jensen, 1996; Pearl, 1988) have an explicit underling probability model that provides a probability that an instance belongs to a class, rather than simply a classification. Support vector machines (Burges, 1998; Cortes & Vapnik, 1995) separate data by creating a hyperplane that is the maximum distance between the data points on either side. It then incorporates only the data points on the margins of the hyperplane border, i.e. the support vector, to define the solution.

Once the system learns the classes, it can very quickly and accurately perform the classification task, much faster than a human can. One of the major shortcomings of this approach as a discovery tool is that it relies heavily on a human operator to provide the examples of each class for the system to learn. The human supervisor decides the classes and the system builds a model that reproduces those given assignments. It is possible that the system will partition the data using some novel and insightful criteria. In addition, the readability of the output can depend on the structure of the model specified. However, it is usually unclear, and often not a concern, how the system determines which example belongs in what class. This also tends to mean that the structure of the model being used is less important than its performance. This lack of transparency in the machine's decisions about how it makes its classification decisions makes it difficult for humans to interpret the resulting class models. In turn, this diminishes the usefulness of classification as an approach to computational scientific discovery, since an explanatory model is preferred.

Clustering tasks share many similarities with classification. Instead of mapping individual data points to predefined classes determined by a supervisor, a clustering algorithm groups similar points together without supervision. Clustering methods can be broadly organized into hierarchical and partitioning methods (Berkhin, 2006; Jain, Murty, & Flynn, 1999). Hierarchical clusters build clusters gradually using a cluster hierarchy, or a tree of clusters. Each cluster node contains child clusters and shares traits with its parent cluster. Hierarchical methods can be further distinguished as agglomerative methods which work from a single point and build clusters or divisive methods which start with all the data in a single cluster then recursively splits the data. Partition techniques divide data into subsets and proceed iteratively, unlike hierarchical clusters which are usually visited once then subdivided. Partitioning methods can also be broadly distinguished as partition relocation methods which iteratively relocate data points between subsets, or density based which try to identify clusters based on areas that are highly populated with data (Berkhin, 2006). The repeated visits in partitioning techniques can produce more refined clusters at the cost of additional computational effort. Relocation partitioning methods include probabilistic like AUTOCLASS (Cheeseman et al., 1988), k-medioids like PAM (Kaufman &

Rousseeuw, 1990), or k-means (Hartigan & Wong, 1979). Density based methods are often used for spatial data and include the algorithm DBSCAN (Ester, Kriegel, Sander, & Xu, 1996) or OPTICS (Ankerst, Breunig, Kriegel, & Sander, 1999). The total number of clusters needed to appropriately separate the data depends on the specifics of the clustering algorithm, the purpose of the data, or could also be determined by the user.

This approach is slightly closer to the kind of discovery task we are interested in because it does not require a supervisor to define the output of the model. The algorithm operates largely on its own to discover an optimal way to organize the data, according to the criteria that defines optimality. This approach can be applied in cases where the data is not very well understood. The algorithm could create a model that organizes the data in a novel way that could be considered a discovery akin to a new taxonomy. Unfortunately the algorithm chooses the clusters based on concepts that have nothing to do with the domain from which the data comes, or at least not commensurate to existing domain theory. Once again the structure of the model is not emphasized, rather performance is what matters. This makes it much more difficult to understand the reasoning behind the resulting clusters. Much like classification, clustering is not ideal for the kinds of discoveries we are interested in for our work.

Association rule models can be considered a type of classification technique but they deserve special attention as a discovery tool because they can produce rules that humans can understand using the language from the domain. Association rule models are often used to describe relationships between categorical items. They are often used in the retail sector where the categories are individual purchases and usually the relationship of interest is simultaneous purchase. Various approaches can be used to determine which associations are most relevant to the relationship of interest. For example, an association rule may be that when milk is purchased there is a 75% chance that eggs will also be purchased in the same transaction.

Association rules can be useful for uncovering empirical relationships between variables. Like clustering, association rule models don't rely on supervised learning to provide examples of rule-following behaviour. However, a user generally has to establish the criteria for what qualifies as a rule worth reporting. The user provides a hypothesis about the data and the algorithm sorts through the data to find evidence to support it. The system outputs a model of the data with a structure that is a set of an association rules. The rule may be very useful and could represent a novel empirical relationship between items. This modelling structure is slightly easier to understand because it is written in terms related to the data being modelled. However, the rules are based on correlations with no underlying theoretical explanation. This lack of a theoretical connection to the domain gives association rule models very weak explanatory power.

Regression analysis is a technique used in data mining (Hand, Mannila, & Smyth, 2001) as well as many other domains (Draper, Smith, & Pownell, 1966; Todorovski, Ljubič, & Džeroski, 2004). In data mining, regression analysis differs from many of the other techniques because it predicts continuous variables rather than discreet categories. The focus is to construct a relationship between the dependent variable and one or more independent variables. This relationship can then be used for prediction and forecasting. In many cases the relationship is expressed as an algebraic function where the dependent variable is a function of the independent variables. Regression analysis is based on statistical relationships between independent and dependent variables. In many domains correlations are useful but they are not sufficient as explanations (correlation is not causation).

Data mining and machine learning methods are used for many applications. The model structures employed by these disciplines are usually abstract which gives them the flexibility to be applied to various domains, often without needing major changes. That helps enable automation, which is an important feature for computational scientific discovery approaches. Unfortunately, the modelling framework used in data mining is usually unrelated to the data being modelled. The lack of connection to the laws and theory of the domain make the models much more difficult to understand by experts in the domain. In addition, the majority of the applications of these methods do not consider discovery in terms that humans can understand to be a primary goal.

2.2. System identification

System identification is a broad topic that involves automating aspects of building a mathematical model of dynamic systems from observed input-output data (Ljung, 2010). This activity is common in several domains so many methodologies have been developed for the various application areas such as aircraft flight controllers (Grauer & Hubbard Jr, 2013) or fault detection (Fekih, Xu, & Chowdhury, 2007). System identification is a step towards discovery compared to data mining techniques because in some cases it places more emphasis on tailoring the structure of the model to the data. The model structure is formed by the components and connections that define the model. In some cases the components within the model correspond to components from the system being modelled. Model structure within system dynamics also refers to its classification, for example linear or non-linear. A model structure that is familiar to domain experts makes the model easier to understand, an important component of discovery.

The task of system identification requires two broad steps. The first step is to identify the structure of the model to be used, then model parameters are fit to specific data in the second step. Both steps have their own distinct challenges. Most system identification work assumes the structural form is known. The structure can be supplied by a domain expert or involve an abstract structure that is very flexible. The focus of the effort then goes into optimizing the parameter estimation stage. Approaches that estimate both structure and parameters share many similarities with the task of computational scientific discovery.

Both system identification and computational scientific discovery are concerned with automating aspects of model building. Computational scientific discovery places an explicit emphasis on the explanatory power of the model as a means to increase understanding. Therefore there is a need for a coherent theoretical framework that underlies the model. System identification, on the other hand, is generally more focused on building practical models to solve the problem of mapping inputs to outputs so that a system can be controlled. In many cases the explanatory power of the model is of less concern than its performance on this task.

A small community within system identification places an emphasis on finding communicable models while retaining a strong focus on applied science, such as the PRET modelling system (Bradley, Easley, & Stolle, 2001). This system automatically constructs ordinary differential equation (ODE) models of given dynamic systems. It is specialized for modelling in the context of system identification where there both inputs and outputs change over time, and in some applications there is a degree of experimental control over the system because it is possible for some of the inputs can be manipulated. For example, in an

aircraft flight controller the control surfaces can be moved and the effect on flight dynamics can be observed. Many approaches to system identification assume a structural form that is provided by an expert because finding an appropriate structure can be a very difficult task. The PRET system restricts the model structure to contain only linear and non-linear ODEs, but even that restricted structure space is infinite.

The PRET system employs various heuristics to limit the structure space. One approach is to use reasoning to check that the structure of the ODE is consistent with a general characterization of the data. For example, a linear ODE model cannot account for chaotic data, thus only non-linear structures would be considered. Another involves organizing the space of all possible models into a hierarchical organization of dynamic systems (Easley & Bradley, 2007). Model building structures for linear hydraulic systems would be different than say linear electronic systems. Once the model structure search space is reduced the PRET system uses a generate-and-test procedure to evaluate ODE model structures, estimating the parameters for each structure against the observed time-series data.

There are many similarities in goals and approaches between the computational scientific discovery and the approach to system identification adopted by the creators of PRET. The PRET system constructs models from existing knowledge and uses language from the domain during the construction process. This gives a degree of comprehensibility to the final models. The main shortcoming of the PRET approach from the perspective of computational scientific discovery is that it is narrowly focused on engineering tasks (Stolle & Bradley, 2007). It is designed for system identification which is concerned with finding a model that captures the required mapping from inputs to outputs, not a model that explains the systems behaviour.

2.3. Equation discovery

Equation discovery is a branch of machine learning that develops methods for automated discovery of quantitative laws expressed in the form of equations (Todorovski & Dzeroski, 1997) and builds upon ideas from empirical discovery (Langley et al., 1987). Equation discovery methods draw from a pool of candidate equation fragments in order to construct equations. In some cases a complete model is composed from several distinct equations. One factor that distinguishes equation discovery from most of system identification is that equation discovery places an emphasis on the discovery of the structure of the model. It is also not designed with a specific discipline in mind. Equation discovery is the first model building technique we have looked at in this review that was developed for the purpose of computational scientific discovery.

Early work in equation discovery uncovered descriptive laws that summarized data. For example, the BACON series of programs (Langley et al., 1987) utilized production rules and was able to discover, among other things, the law for the conservation of momentum when given data such as the initial and final velocities of objects and their masses. Other early work exploited dimensional analysis to discover the equations of motion (Kokar, 1986). Discovery of descriptive laws such as these usually occurs early in the development of a field before complete theories have been established. These early equation discovery systems were able to rediscover historical human discoveries. They provided evidence that computational scientific discovery was an achievable goal.

Equation discovery has also been used successfully to model regression relationships (Todorovski et al., 2004). Instead of using statistical relationships to build a regression equation, it is constructed from

polynomial components. This approach can be useful in certain domains or on data where the statistical assumptions that underlie regression do not hold.

Later equation discovery systems expanded the approach with the ability to model dynamic systems expressed as ordinary differential equations (ODEs). Much like in system identification, the problem of navigating the model structure space remains a central issue. Equation discovery is intended to be applicable to any domain so it cannot use many of the assumptions that applied in system identification. The space of possible model structures, even when restricted to ODE models, is infinite. Two approaches are generally used in tandem to address this problem. The first is to incorporate some form of prior knowledge to carefully define the space of ODE structures to reduce the number of possible equations without removing relevant possibilities. The other approach is to develop heuristic search algorithms that more efficiently search through the space.

Grammar based equation discovery (Todorovski & Dzeroski, 1997) is an approach that uses knowledge to constrain the space of ODE structures. This method organizes ODEs using a context-free grammar, where individual equation structures are equivalent to sentences. The grammar rules constrain the complexity of individual polynomial terms that appear in an equation as well as the complexity of the entire equation structure. Domain knowledge needs to be used to develop the grammar rules for each application thus giving equation discovery the flexibility to be applied to any domain. The grammar based approach has been successfully applied in domains such as ecology (Ivanovska, Todorovski, Debeljak, & Džeroski, 2009; Todorovski, Džeroski, & Kompare, 1998).

The discovery system CIPER (Džeroski & Todorovski, 2008) builds polynomial equations that are linear in the parameters. This allows it to use linear regression to estimate parameters. It is also able to use partial models as the prior knowledge to reduce the search space. Partial models have some of the model structure defined meaning that some of the polynomial terms in each equation are provided ahead of time. These partial equations provide a starting point for model structure search. The CIPER system also incorporates knowledge about the typical forms of the polynomial terms that appear in that domain to further reduce the space. A relatively simple search heuristic based on the minimum description length for equation complexity is then used to help narrow the search space. The CIPER system was able to successfully reconstruct a known biochemical reaction network when provided with a partial model as a starting point. However it was unable to find an adequate model in the given time constraints when starting with no prior knowledge, demonstrating the value of reducing the size of the space.

Recent work in equation discovery by Brunton et al. (Brunton, Proctor, & Kutz, 2016) leverages sparseness to make the identification problem more tractable. Sparseness reflects the idea that the polynomial equation will usually have a few terms while the space of possible expressions is very large. Similar to the CIPER system, this approach also uses regression to fit coefficient values to a set of polynomial expressions. In order to take advantage of using regression for sparse problems, they developed an equation discovery algorithm that incorporates the least absolute shrinkage and selection operator (LASSO) (Tibshirani, 1996). LASSO uses a combination of subset selection and ridge regression to select coefficients. Subset selection involves removing irrelevant regressors in a discrete process while ridge regression removes regressors by iteratively reducing some regressor coefficients to zero. The performance of this algorithm is very promising, but the approach suffers from the same fundamental issue in equation discovery because it is employed as an equation discovery technique.

However, there is no reason why it could not be used in a process modelling approach. It is a promising avenue for future research.

Another approach to managing the equation search space is to use a more sophisticated search strategy. Evolutionary algorithms are one such search strategy that has been used for equation discovery (Cornforth & Lipson, 2013; Iba, 2008). Most evolutionary algorithm techniques revolve around the idea of selecting a random sample of candidates from the total pool of possibilities, evaluating their fitness, then retaining those candidates with the highest fitness scores for further testing. These high fitness candidates are sometimes combined in clever ways with other high fitness candidates or slightly altered, or mutated, then tested against the parent candidates. Developing the criteria for fitness as it pertains to the task of equation discovery is a challenge.

Evolutionary algorithms in equation discovery are generally used to find a model structure but they can also be used to find parameter values (Iba, 2008). Usually more traditional methods are incorporated to estimate model parameters. An algorithm that can identify promising candidates in a domain agnostic way would reduce the need for human expertise for computational discovery. Discovery systems that use this approach would be easier to configure because they don't require as much prior knowledge about the domain. This is the approach taken by (Schmidt & Lipson, 2009) in order to find equations that express natural laws from experimental measurement of the motion of pendulums and other mechanical systems. Their system focuses on uncovering functional invariants within the data, similar to some of the earliest discovery systems. An invariant function remains constant across a set of inputs. For example, the total energy of an orbiting body in space is the constant sum of its kinetic and potential energies, regardless of its current speed or position.

The space of candidate invariant functions is very large. Noise that is inherent in empirical data requires that approximate invariants be considered as well. Their approach used an evolutionary algorithm to select for more promising invariant equations. The use of invariants is important in many areas of physics but it can be challenging to apply those same concepts in other domains. Later work applied the same concept of genetic algorithms to the more general problem of constructing ODE models for any domain as well as the challenging problem of allowing for hidden variables (Cornforth & Lipson, 2013).

ODE models created using equation discovery are built from a pool of candidate equation components. The components are usually in the form of a polynomial expression. The model structure space is infinite so it is necessary to reduce its size using prior knowledge. The resulting equations are expressed in a mathematical format that in theory should be understandable to a domain scientist. However, the polynomial components that end up in an equation together are only loosely affiliated because they lack a theoretical basis related to the domain being modelled that could explain why they must appear in an equation together. Practically speaking, the various components appear in the equation together because they followed a grammar-like rule, or were rated highest by the genetic programming fitness function. The lack of a domain specific reasoning behind an equation discovery systems choices of which fragments to include can make it difficult for domain experts to understand the model. The equations serve as descriptive summaries of the data, not explanations of it.

2.4. System dynamics

System dynamics is a modelling methodology that also focuses on improving understanding of the system through the construction of models. It was initially known as industrial dynamics and was

originally developed for business and industrial purposes (Richardson & Pugh, 1981) and has since been applied to other domains including healthcare (Brailsford, 2008) and hydrology (Khan, Yufeng, & Ahmad, 2009). The models are usually intended to be a decision making aid that addresses a specific policy problem.

System dynamics emphasizes a theoretical framework that underlies each model created using this method. As previously pointed out, equation discovery does not emphasize any unifying principles thus models created using that technique can be more difficult to interpret. Models in system dynamics are constructed under a theoretical assumption that dynamic systems are best understood as feedback loops. A feedback loop means that the current value of a variable affects its future value. The feedback can either be positive which reinforces the existing direction of change, or negative which slows or reverses the direction of change. It is the job of the human modeller to determine which variables are of interest and the feedback loops that are relevant.

An initial conceptual model is built using causal-loop diagrams that identify the relevant variables and the type of feedback loop, either positive or negative. This is often accomplished by using a stock and flow notation. Stocks represent the quantity of a variable, and flows cause changes in the quantity of those variables. The notions of stocks and flows usually need to be translated into the terminology of the target domain by the modeller. For example, in population dynamics a stock could be a population and a flow might include anything that changes that population's quantity such as birth, death, or migration.

The stock and flow diagram can initially be a qualitative representation that only posits the direction of flows and the stocks they influence. In most cases the modellers will investigate details about the flows and their influences to make them quantitative. Understanding the quantitative behaviour of flows opens up the possibility of creating ways to manipulating the flows to better control the dynamics of the system. If the system dynamics model is encoded as stock and flow diagram with the flows quantitatively defined, it can be translated into an ODE model that makes testable predictions. Various equation fragments that make up the ODE correspond to a stocks and flows in the diagram.

System dynamics represents a modelling paradigm that is theoretically unified through its representational assumption of feedback loops. Once quantified as stocks and flows, the model can be expressed as a set of ODEs. Each component of the ODE can be traced to its role within a feedback loop, thus providing an explanation of its purpose in the model. The main drawback of this approach for computational scientific discovery is that very little system dynamics work is concerned with automation of this process. Human labour is required to identify the relevant variables, their causal-loop diagrams, and then refine those broad notions into an executable model. In order to automate aspects of this process, its theoretical assumptions need to be defined more specifically so that it can provide domain specific explanations.

2.5. Qualitative reasoning

Qualitative reasoning about physical systems, also known as qualitative physics, uses representational assumptions based on how human beings reason about physical systems. This can allow qualitative reasoning more flexibility than system dynamics, which restricts the qualitative representational assumptions to only feedback loops. Qualitative models focus on the qualitative behaviour of variables, i.e. is it going to increase or decrease, which is often what is of most interest. It is motivated in part by

the observation that human beings seem to be able to make useful predictions about physical systems without resorting to solving and simulating differential equations (Forbus, 2014).

One of the goals of qualitative reasoning research involves understanding the methods that human beings use to solve physics problems. This knowledge can then be used to improve the performance of AI systems such as software for robots in unconstrained environments or expert systems (Hunt, Lee, & Price, 1993). Using reasoning techniques that humans use can also help the systems output be easier for humans to understand. This overlaps with the goal of computational scientific discovery which is to develop systems that produce human level discoveries using terminology that humans can understand.

Some of the advantages of qualitative modelling include ease of understanding, a reduction in the sensitivity to noisy data, and the elimination of numerical parameters (Garrett, Coghill, Srinivasan, & King, 2007). Any empirical data that is collected will have noise but qualitative representations are less sensitive to noise. Qualitative models also do not have parameters that need to be estimated which greatly reduces the computational cost compared to something like a differential equation model that would require system identification techniques to find parameters. Qualitative reasoning methods have been applied to simulate and identify metabolic pathways in the domain of biology (King, Garrett, & Coghill, 2005).

Many approaches to qualitative reasoning incorporate aspects of the formalized qualitative mathematics developed by Kuipers (1986). It involves identifying landmark values for observed variables and grounding reasoning on those landmarks instead of on continuous values (Hunt et al., 1993). In many cases, quantities are discretized into an ordinal set of possible values. For example, the values can be as simple as positive, zero, or negative. This can help partition the behaviour of the system into distinct states. Transitions between different states occur when the values change. Qualitative models often do not produce a single future state for the system, instead they predict an envisoinment of the future states of the system which is a collection of all the possible states and their transitions.

Qualitative reasoning approaches also tend to emphasize the compositionality of their concepts of physical systems (Bobrow, 1984). Compositionality means that each component in the system has some behaviour and the dynamics of the system as a whole emerges from the interaction between the various components. The systems behaviour must be derivable from the structure of the system. Compositionality can give the models explanatory power, provided the components in the model explicitly correspond to components in the real system. One example of this is in circuit diagrams where the model components such resistors and capacitors correspond directly to components on a physical circuit board. Qualitative process theory (Forbus, 1984) introduces the notion of a physical process as the organizing component for physical systems. It proposes that the interaction between objects in a system be defined as processes. The dynamics of the system are the result of the interaction of various processes.

A process defines a relationship between objects in a system. For example, in chemistry a simple chemical reaction can be represented as a process where one type of substance changes into another. Qualitatively that could be represented as one substance reducing to quantity zero while another increases. Processes have conditions under which they are active, relations that describe what the process does, and influences that determine what is affected by the process. Processes can start and stop when the properties of objects change, i.e. when an objects landmark value changes it can activate

or deactivate a process. The process itself is a theoretical construct that connects to a larger domain theory. A single process can serve as the explanation of a very simple kind of interaction within the system. The dynamics of a complex system can be explained in terms of the interaction of many of these simple processes. A model composed of qualitative processes provides a framework for explaining the system.

Compositional modelling (Falkenhainer & Forbus, 1991) is an idea related to qualitative reasoning that organizes modelling tasks in order to automate certain aspects. It involves using modelling assumptions to automatically decompose domain knowledge into a library of semi-independent modelling fragments. Those fragments can then be combined into a model that describes some behaviour of the system. Different sets of assumptions will produce different fragment libraries that would be appropriate to address modelling questions relevant to the assumptions. Qualitative reasoning methods are needed to represent and reason about the modelling assumptions to choose libraries that are relevant to a particular query. For example, Nakak & Joskowicz (1996) demonstrated the various heuristics and additional reasoning that were required to apply a compositional modelling framework to electrochemical devices.

Qualitative process theory can define models that are gross approximations of the real system. Qualitative mathematics can then be applied to make predictions about the future states that are possible for the system. This approach does not require exact data to produce results, and the analysis can be very fast in some cases. Compositional modelling techniques can be applied to qualitative process theory to provide a way to automate aspects of model construction. The process based representation provides explanatory power to the models. One of the main drawbacks of qualitative methods are that the results can be ambiguous and no useful conclusions can be drawn from them. In that case, a more detailed model such as a differential equation model, becomes necessary to resolve ambiguity.

2.6. Quantitative process models

Qualitative processes as discussed in a previous section provide an explanatory framework that can be useful for computational scientific discovery tasks. However, it is not intended to provide the level of detail to represent exact values for continuous variables. In order to represent dynamic systems with continuous variables a quantitative process theory is needed that extends the ideas of qualitative processes. A quantitative model can provide additional detail and resolution that can appear in scientific models. An automated approach to constructing quantitative processes models has been developed and introduced as the task of inductive process modelling (Langley et al., 2002).

The processes in the framework used in for inductive process modelling is intended to capture a relatively simple relationship between variables in the system, expressed using mathematical notation. The processes themselves are unobserved, but their actions are responsible for changes in the values of the variables in the system, which is observable. Processes are still compositional in nature and the dynamics of the system emerge from the interaction between individual processes. The relationship between variables in a process can be expressed as a differential equation where the relationship extends over time, or an algebraic one that represents an instant effect. Instant effects are an approximation of events that take place quickly relative to the time scale of the measurements. A set of processes define a model. The model can be expressed using differential equation formats that are common in many science and engineering domains.

Processes are defined in two different forms in this framework: generic and instantiated. Domain knowledge is encoded as generic processes. A generic process accounts for a relationship between variables that can appear in a domain. Generic processes must include a name, a statement about which variables that can be involved, and a set of equations that specify its effects. It can also include a set of conditions for when the process is active. In most cases the generic process definition will include some unknown parameters that can be determined with available data. The parameter definition within the generic process can include constraints on the allowed value of the parameters.

Then notion of exponential growth over time provides a simple example of a generic process. In differential notation the growth of a variable x may be expressed as dx/dt = k * x. When that equation is integrated with respect to time it takes the form of $x(t) = x_0 * e^{kt}$ where x_0 is the initial quantity, k is the growth constant, t is time, and e is Euler's constant (approximately 2.718). It captures a relationship between the current and future value of the variable x. A process of that form can appear in many different domains including population dynamics, radioactive decay, and economics. When this generic process, which expresses a relationship for some variable labelled x, is applied to a specific domain it becomes a process instance in that domain, where x would become a specific population, or a quantity of radioactive material, or some other specific quantity of the domain. A single generic process can result in several different process instances. For example, in some ecosystem there may be there may be two different populations, x and y, both of which could be influenced by an exponential growth process. Thus the single exponential growth process would create one process instance for each variable. The parameter *k* is unknown, has unique value for each instance, and would need to be fit to the available observed data.

Variables from the data can be directly substituted for the generic variables defined in the generic process until all possible instances are created. Each unique combination of process instances represents a different un-parameterized model structure. Process instances are unique within a single model so the number of possible model structures grows as the factorial of the total number of process instances. The exponential growth of the model structure search space must be controlled in order to make the search tractable.

Type constraints on the variables are included as part of the framework in order to help reduce the number of possible process instances. These allow a modeller to define that certain variables in the data be associated with types, and generic processes to be restricted to only influence certain types. For example, the exponential growth generic process can be restricted to apply to only variables that are of type organism. Going back to the previous example, if x were an organism type, but y was a nutrient type, our exponential growth generic process can be restricted to only apply to organism types, and only one process instance would be created instead of two.

The model structure needs to be evaluated in order to know if it can account for available observed data. The set of process instances that define a model structure can be compiled into a differential equation model. This format allows it to be simulated using existing techniques for differential equation models. There are usually unknown parameters that need to be found. The amount of error between the model prediction with optimized parameters and the observed data provides an evaluation metric. The task at this stage is that of system identification, finding the values for the parameters in a differential equation model that minimize the error with respect to the given data.

The parameter estimation stage can be very computationally intensive. In many cases the model contains multiple parameters and the parameters are embedded within a set of coupled differential equations that must be simulated to calculate error. Gradient descent methods that estimate parameters require multiple error calculations which means that the differential equation system must be simulated multiple times per structure. In addition, the parameter error space is often non-linear which means that a global optima cannot be guaranteed. In order to increase the coverage of the error space in the hopes of finding the global optima in gradient descent methods, multiple restarts are required. This effort must be repeated for each model structure.

The result of the search through the structure and parameter spaces can produce one or more models that satisfy the criteria of an acceptable model. These criteria can include things like the amount of error between model prediction and observed data. Some error is expected due to noise in the data. The search could fail to find acceptable models if the generic processes do not contain the necessary information to generate process instances that can be combined to fit the data. In this case, parameter estimation should fail to find a model that matches the data well. It is also possible that an appropriate structure is found, but parameter estimation fails to find the global optima and ends up returning poorly fitting parameters.

The resulting models are composed from a sets of process instances and their corresponding parameter values. Each process model can compile into a differential equation and be simulated to make predictions about future behaviour. Organization of the differential equation model into processes provides the explanatory power needed to support discoveries that was missing in equation discovery approaches. Each process that appears in the model corresponds to a domain theory that was encoded as a generic process. Domain experts should be able to understand the role that each generic process plays in the domain, thus every component in the model equations can be attributed to a particular process.

The quantitative process based framework has been applied in several domains including ecology (Asgharbeygi, Langley, Bay, & Arrigo, 2006), hydrology (Bridewell, Langley, Todorovski, & Džeroski, 2007), and biochemistry (Langley, Shiran, Shrager, Todorovski, & Pohorille, 2006). In addition, several advances have been made since its introduction. In domains such as ecology it can be a challenge to collect continuous data over long time periods or measuring devices could be unreliable. Techniques for learning process models with missing data were developed to address those issues (W. Bridewell et al., 2006). Structure search grows exponentially so later developments introduced new methods to reduce the structure search space. One method involves learning clauses that characterize accurate and inaccurate models (Bridewell & Todorovski, 2007). This learned knowledge can be used to bias the search towards models that could be more accurate. Another development incorporates domain knowledge in the form of explicit structural constraints in the generic process definition (Bridewell & Langley, 2010). These constraints include: necessary, always-together, at-most-one, and exactly-one. The constraints expresses a relationship between generic processes that reduces the size of the structure search space. The notion of entities was another advancement that plays an important role within this process model framework. An entity is a collection of the variables and constants that define its state (Bridewell & Langley, 2010). They are the actors and receivers of action in the system. Entities and variables become equivalent when there is only one variable that defines the state of an entity. Explicitly defining entities is a natural way to group variables and parameters in way that more closely resembles how scientists think about real systems (Borrett et al., 2007).

Despite this progress, the current approach to inducing quantitative process models faces three fundamental challenges. Although process models are compositional, they can only be evaluated as complete models. This approach will not scale to complex because the model structure space grows exponentially with the number of variables. It is unclear if it is possible to identify more promising model components without putting them into a complete model first. A second issue is the computational cost fitting unknown model parameters to the data, which is required to evaluate each model structure. The approach to fitting parameters taken by current systems involves a gradient descent algorithm that requires simulating the model across the time span of the data. Each step in the gradient descent requires a new simulation. This means that each model structure requires many repeated simulations to fit the parameters. The final issue is that the parameter space is non-linear and may contain local optima. Techniques that more thoroughly search the parameter space for a global optima require additional restarts which further increases the computational burden. The recent induction system SC-IPM (W. Bridewell & Langley, 2010), 99.99% of the CPU time was spent running simulations for parameter estimation.

2.7. Concluding remarks

Table 1 provides a summary of the approaches discussed in this review. The predicted variable type is what the model predicts. It can be discrete which means that the model predicts distinct separate states, versus continuous outputs that are numerical values. Our approach to computational scientific discovery focuses on being able to predict continuous variables. Input data type is about whether the approach typically deals with dynamic inputs where the temporal order that the data is received matters, or only deals with static data. Automation of structure and parameter identification indicates whether the approach focuses on automating either of those modelling tasks. Most data mining, system identification, and system dynamics approaches do not search for the appropriate model structure and instead rely on the user to provide one. Parameter estimation is almost always automated to some degree because it can involve a solving computationally demanding minimization problem. System identification is by and large a discipline dedicated to the problem of parameter estimation for dynamic systems. The model interpretability rating is my opinion on the degree of relation between the model structure and the domain that it aims to model. A direct relationship between model structure and the modelled system facilitates explanations because domain experts should be familiar with the concepts that make up the model. Data mining methods provide the worst levels of interpretability because the approach does not focus on that. Processes, both qualitative and quantitative, provide the best interpretability because they aim to capture relationships between objects within the system using a componential arrangement. The quantitative version is best suited for our approach to computational scientific discovery because it predicts continuous variables rather than discrete states.

Table 1. Summary table of approaches related to computational scientific discovery

	Predicted variable type	Typical input data type	Automatic structure identification	Automatic parameter estimation	Model interpretability rating (1-5)
Data mining					
Classification	Discrete or continuous	Usually static	no	yes	1
Clustering	Discrete or continuous	Usually static	no	yes	1
Regression	Continuous	Static and dynamic	no	yes	1
Association rules	Discrete	Usually static	no	yes	2
Equation Discovery	Continuous	Dynamic	yes	yes	3
System Identification	Continuous	Dynamic	Usually not	yes	4
System Dynamics	Continuous	Dynamic	no	yes	4
Qualitative Processes	Discrete	Dynamic	yes	n/a	5
Quantitative Processes	Continuous	Dynamic	yes	yes	5

Qualitative processes provide the most appropriate representation for developing our computational scientific discovery tool. In order to address the scaling issues facing the current best quantitative process modelling approach, a new formulation for quantitative processes is needed. In the following section, I introduce the concept of a rate-based process along with some of the concepts that directly lead to its development. I provide more detail about the SC-IPM system scalability limitations and how the new formulation addresses the scalability concern while still maintaining the explanatory power and flexibility of a process representation.

3. An improved parameter estimation algorithm

The review of the literature in chapter two provides evidence that a quantitative process based modelling framework is the most promising for the development of a computational scientific discovery system. The latest development along this front is the discovery system is the system called Structurally Constrained Inductive Process Modeller (SC-IPM) (Bridewell & Langley, 2010). The first section of this chapter describes the approach taken by SC-IPM and how lessons from that approach lead to the development of a new algorithm to estimate parameters. The new algorithm addresses some of the scaling issues that hindered the SC-IPM system. Section two goes into more detail about the development and testing of the new regression-based algorithm used to estimate parameters.

3.1. The problem with SC-IPM

The SC-IPM system uses a framework based on quantitative processes. The system is designed to be a tool to assist scientists in making discoveries by automating the model structure and parameter search tasks resulting in a model that both accurately accounts for the data and explains the behaviour. The structure of a process model defines what processes appear as part of the model and defines relationships between objects in the system. Parameters are numerical constants that appear as part of the model. Background knowledge provided by the domain expert is incorporated into this framework through a generic process library. The library specifies all the ways that the objects within a particular domain can interact and often connects to domain theories and taxonomy. As a simple example, in the domain of organism ecosystems, there could be a predation process that involves one species consuming another that causes a decrease in the prey population and an increase in the predator population. The generic processes are made into specific instances of a process when applied to actual data. For example, the data may contain variables *x* and *y* and one of the specific process instance created from the generic process relationship is that *x* consumes *y*. Process instances form the components of the model structure and their interaction creates the dynamic behaviour of the system.

A complete model is a combination of processes instances and their associated the parameter values that accounts for and explains the given data. The process model can be expressed as a set of differential equations that can be simulated to make predictions about the future behaviour of the system. The processes within the model form the basis for explaining the function of the model, which can be traced back to the generic process library which connects to domain theory and taxonomy. The connection to domain theory through the processes provides the explanatory power necessary for computational scientific discovery.

Figure 1 shows an example model output from SC-IPM where the processes are identified with labels. The system being modelled is a predator prey eco system with the variable *Aurelia* playing the part of the prey species and *Nasutum* as the predator. The first equation states that the rate of change of the prey species involves the difference between “exponential-growth” and “grazing” with the former having a positive coefficient and the latter being negative. This translates to the idea that the prey species rate of change is positively influenced by growth and negatively influenced by being grazed upon, with the net rate of change of the prey species being dependent on the interaction of these two processes. Similarly, the second equation states that the rate of change of the predator species *Nasutum* involves the difference between “exponential loss” and “grazing” with the former being negative and the latter having a positive sign. This translates to the idea that the predator species is negatively influenced by a loss mechanism but is positively influenced by grazing. The grazing process

captures the idea that grazing both reduces the population of the prey species while simultaneously increasing the predator species population.

MODEL-58 :: Score = 5.4244d-1

Equations

$$\frac{\mathrm{d}[conc_{\text{aurelia}}]}{\mathrm{d}t} = \underbrace{(growth\_rate \cdot conc_{\text{P}})}_{\text{exponential-growth(P:\{aurelia\})}} + \underbrace{(-1 \cdot grazing\_rate_{\text{G}} \cdot conc_{\text{G}})}_{\text{grazing(G:\{nasutum\}, P:\{aurelia\})}}$$

$$\frac{\mathrm{d}[conc_{\text{nasutum}}]}{\mathrm{d}t} = \underbrace{(-1 \cdot loss\_rate \cdot conc_{\text{G}})}_{\text{exponential-loss(G:\{nasutum\})}} + \underbrace{(assim\_eff_{\text{G}} \cdot grazing\_rate_{\text{G}} \cdot conc_{\text{G}})}_{\text{grazing(G:\{nasutum\}, P:\{aurelia\})}}$$

$$grazing\_rate_{\text{nasutum}} = \underbrace{gmax_{\text{G}} \cdot (conc_{\text{P}}{}^{alpha})}_{\text{generalized-gause(G:\{nasutum\}, P:\{aurelia\})}}$$

Parameters

nasutum :: gmax = $4.1827d-2$
nasutum :: assim_eff = $1.0560d-1$
generalized-gause(G:{nasutum}, P:{aurelia}) :: alpha = $9.9990d-1$
exponential-loss(G:{nasutum}) :: loss_rate = $9.3127d-1$
exponential-growth(P:{aurelia}) :: growth_rate = $2.7154d+0$

Figure 1 The model structure as reported by SC-IPM when given data of two interacting species in a predator-prey ecosystem

The model shown in Figure 1 is a process model expressed in differential equation format. This model was one of the best fitting models found when SC-IPM was presented with empirical data of a predator-prey ecosystem described in (Bridewell & Langley, 2010). Trajectories that result from simulating this model are shown in Figure 2 and provide a synthetic data set that will be used to further to evaluate the performance of the SC-IPM system. The synthetic data set provides many advantages for testing over using the empirical data including a known search target, the ability to generate a data set of any arbitrary size, and a lack of noise. The synthetic data will help highlight some of the undesirable outcomes of the parameter search algorithm incorporated into SC-IPM. The system should have no trouble recovering the exact model of this perfect data when provided with the appropriate generic process library. Parameter estimation seems to be the main difficulty encountered by the SC-IPM system so it is provided with the known model structure, which eliminates the need to search for one, in order to focus the search effort solely on parameter estimation.

The parameter estimation routine involves a gradient descent search. An initial set of random parameter values are chosen and used to run a simulation. The error between the simulation output and the target data is computed and used as part of an objective function to minimize the sum of squared error between the predicted data and the target data. An error gradient with respect to each

parameter is computed which gives an estimate of the effect of changing a parameter value on the overall error. The parameter values are adjusted in a direction that will drive the error to a smaller value then those values are simulated again and a new error is computed. The steps of adjusting parameters and simulating are repeated until the gradients are approximately zero. That condition indicates that the error has reached a minimum. In many cases the error space will contain many local minima so the parameters must be re-initialized to random values and the process repeated to increase the chances of finding the global minimum. The increased chance to find the set of parameters with the smallest error comes at the cost of additional computational effort. The effect of many local minima requiring additional restarts will prove to be one of the primary causes of SC-IPMs inefficient search.

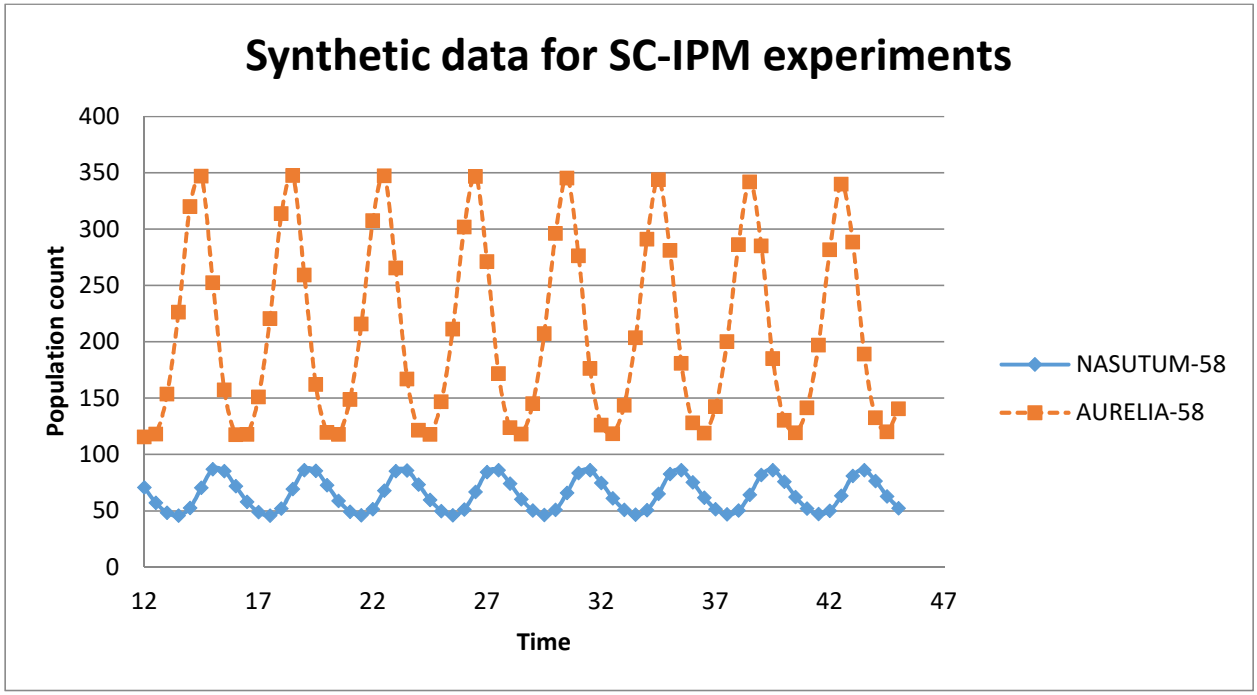


Figure 2 Synthetic data of a two variable predator prey system.

Figure 3 shows the resulting distribution of model scores when SC-IPM is provided with the model structure and asked to run parameter estimation 2500 times with no random re-initializations. Note that both the vertical and horizontal axes have discontinuities to try to capture the extreme values without removing all of the detail. The reported score is a mean squared error measurement, it is calculated by first taking the mean value of the data and calculating the error of each data point relative to the mean value. It is a ratio of the reported error over the error of the mean model. The mean model can be thought of graphically as drawing a straight horizontal line through the middle of the data. Practically this measure states that a score of 0 is a perfect fit and a score greater than one is worse than a model that just predicts the average value of the data. Results from the 2500 estimations showed that over 1700 different unique scores, rounded to 3 decimal places, were reported and five out of the 2500 attempts resulted in the correct set of parameters, i.e. the parameters that were used to create the data. All results that were not the correct set of parameters scored 1.0 or higher, which indicates their fits were equivalent or worse than a mean value guess.

The experiment involved a single parameter estimation loop which gives some evidence of the minimum number of local minima in the parameter space. Several conclusions can be drawn from the results of this experiment. The first is that the parameter space contains a large number of minima. The parameter search algorithm appears to be inherently unreliable with this data set because the random initial conditions are unlikely (5 out of 2500 tries) to find the global minimum. The algorithm also doesn't seem to find models that are close because the second best reported model score was greater than 1.0 and thus worse than a mean model.

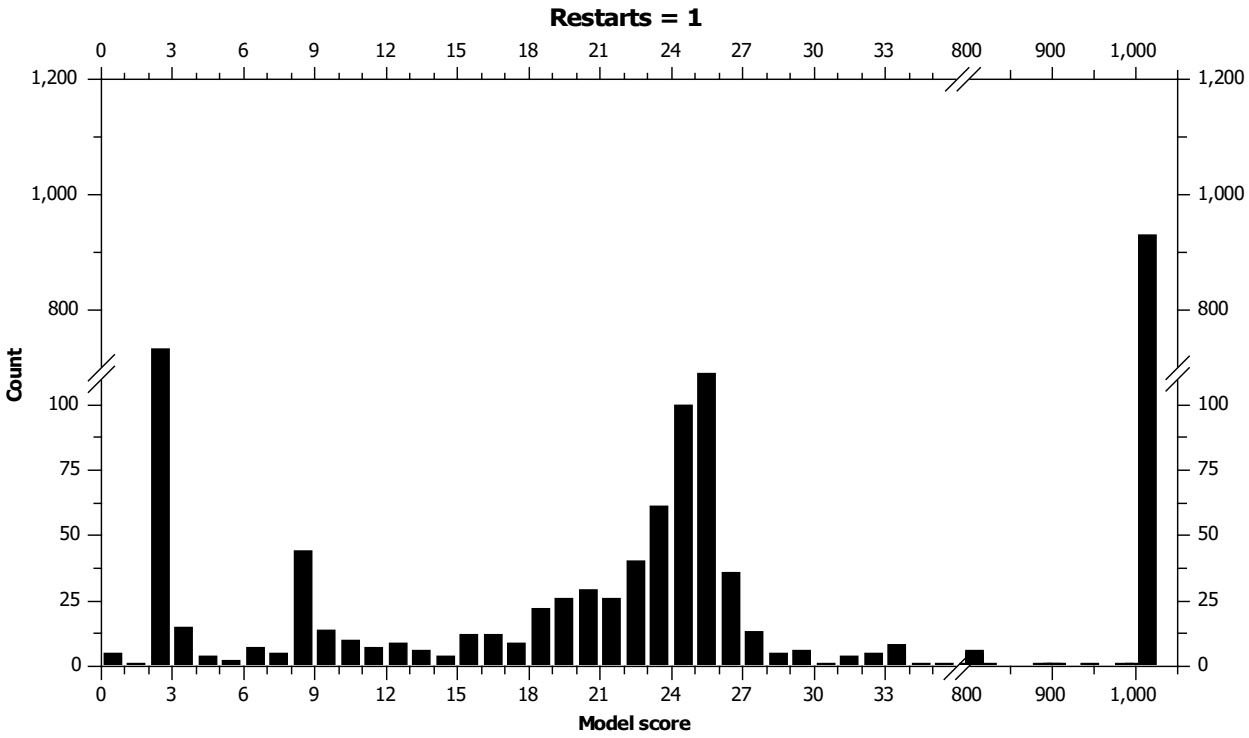


Figure 3 Distribution of model scores reported by SC-IPM when asked to perform a single parameter search loop.

Table 2 shows the results from an experiment to try to determine the success rate of the program as a function of the number of parameter estimation retries. Parameter estimation restarts refers to the number of times the parameter estimation algorithm is restarted with random initial values. The number of model finding attempts is how many times SC-IPM returns the best of parameters it found from those parameter estimation restarts. Successes are counted when the best set of parameters returned are the correct parameters. With 250 parameter estimation restarts, the system returned the correct set of parameters on 10 out of 30 runs. This suggests that if you provided the system with the correct model structure and allowed it to restart parameter estimation 250 times, SC-IPM would have about a 30% chance of finding the correct parameter values. At 500 parameter estimation restarts, it was able to find the correct parameters on 22 out of 30 attempts. At 1000 parameter estimation restarts it was successful 27 out of 30 times. A second, similar data set showed a success rate of 29 out of 30 with 1000 restarts. This provides evidence that on this particular data set if you configure SC-IPM to run 1000 parameter estimation restarts, odds are about 90% that it will return the correct parameters. An ordinary desktop computer circa 2013 took approximately two minutes to run through

1000 parameter estimation retries. This for a single model structure that only has two observed variables and five parameters. This effort must be repeated for every model structure in the space. Presumably the parameter estimation algorithm will take longer where models contain more parameters and more variables and there is also no guarantee that the levels of reliability for this data set will be consistent in other settings.

Table 2 Results from initial reliability experiments of the SC-IPM parameter estimation algorithm when given the model structure and asked to find parameters

Parameter estimation restarts	Model finding attempts	Success count	Success %
1	2500	5	0.2
10	250	6	2.4
20	125	5	4.0
50	50	4	8.0
100	25	9	36.0
250	30	10	33.3
500	30	22	73.3
1000	30	27	90.0

Careful examination of the model output in Figure 1 suggests another issue with this framework. The third equation in our model expresses an algebraic process for the grazing rate. This process is mathematically substituted as a variable into the other equations. In this particular model, the exponential parameter *alpha* ends up being assigned to 0.9999 which satisfies its upper bound of being less than 1 but still approximates to 1.0. That means that this model structure is approximately identical to another structure with no exponential term: $gmax \times conc_p \approx gmax \times conc_p^{0.9999}$. The parameter values in this model instance demonstrate how parameters can allow for a structural degeneracy, where two supposedly different structures are approximately equivalent. That can make it more difficult to understand the model because the parameters it proposes are effectively nulled out, which essentially converts a process into another process that is supposed to be distinct. The structural degeneracy problem is a relatively minor issue compared to the scaling problem as it could potentially be addressed with some additional constraints on the parameter values.

3.2. The development of a regression based parameter identification algorithm

The unreliability coupled with the CPU effort required for the gradient descent based parameter estimation calls for a different approach. An insightful reformulation of the process framework helped lead to the development of a regression based algorithm. The reformulation allows the differential equation representation of a process model to be thought of as a set of independent regression equations. This section provides the evidence that a regression based algorithm can identify parameters that that are part of a process model that is able to reconstruct the original data. This method provides a way to estimate parameters without the need for repeated simulation which addresses the most critical shortcomings of the approach used by SC-IPM.

The section describes the reformulation of the representational framework used by SC-IPM and how it changes from a differential equation simulation problem into one that looks like it could be solved using

regression. Next comes a discussion of the assumptions that are required and the methods that are used to solve the differential equation representation using regression techniques. After that comes a discussion on some of the sources of error for this algorithm including estimating derivatives and simulation. The section closes by discussing evidence from synthetic data sets that suggest a regression based parameter estimation algorithm can successfully identify parameters and reject incorrect model structures. This regression algorithm only deals with parameter estimation, which is only one step of the computational discovery process as it assumes that the model structure is already known. The next chapter discusses the implementation of these ideas into a larger framework that performs all steps required for computational scientific discovery.

3.2.1. Process based representation that emphasizes regression

The model display in SC-IPM produces an output of the form shown previously in Figure 1. In that notation the model equations are expressed as differential equations with the processes indicated underline bracket notation. Identifying the processes is an important component of the explanatory power of a process model. One of the other advantages of this way of expressing a process model is that differential equation format is a familiar way of expressing dynamic systems in many domains. A different notation would be better suited to clarify how a regression based algorithm can be used to identify a parameters.

Table 3 A conversion table for names of objects in the SC-IPM model translated to clarify how a process model can be expressed in a format that can be solved using multiple regression. Part (a) shows the name of the parameters as defined by SC-IPM along with the new parameter name as it appears in a regression equation. Part (b) demonstrates how a process model can be expressed in a form that can be solved using regression. Part (c) gives the definition of the process rates so they can be calculated from the observed concentrations.

(a)

SC-IPM Parameter Name	New Parameter Name	Numerical value
nasutum::gmax	b	- 0.041827
nasutum::assim_eff	c = b*assim_eff	0.004417
generalized-gause::alpha	Round to 1.0 and drop	0.9999 ≈ 1
Exponential-loss::loss_rate	d	-0.093127
exponential-growth::growth_rate	a	2.7154
$conc_p$	$conc_N$	-
$conc_g$	$conc_A$	-

(b)

$$\frac{dA}{dt} = a\,P_1 + b\,P_2$$
$$\frac{dN}{dt} = c\,P_2 + d\,P_3$$

(c)

$$P_1 = conc_A$$
$$P_2 = conc_A * conc_N$$
$$P_3 = conc_N$$

The differential equation format of a process model can be written in such a way that it becomes a very familiar looking regression problem. The equation format output by SC-IPM from Figure 1 can be condensed into the equations shown in Table 3(b) with the help of parameter translations in Table 3 (a). The parameters listed under the model are reassigned to single letter names. The parameter called *alpha* appears as an exponential and is rounded off to 1.0 then dropped because an exponential to the first power is redundant, so that parameter does not appear in this reformulation. The processes are

renamed to P_1, P_2 and P_3, and their mathematical expressions are defined in Table 3 (c). The models shown in Figure 1 and Table 3 (b) are equivalent. The formatting of the equations Table 3 (b) is intentionally indented to line up the processes vertically and make it clear that P_2 appears in both equations. The reformulation is intended to make it clear that each derivative is a linear combination of processes that are each multiplied by a single coefficient. The reformulation provides a starting point for discussion in the next section about how to solve these equations using regression.

This reformulation does not make a strong attempt to preserve the notion of processes at this stage. Processes still play a role in the organization of the equations but they do not require a formal definition yet because they aren't being used for induction so they do not need to consider a generic process definition. This notation serves as a stepping stone between the previous process framework used in SC-IPM, and the rate-based process framework that will be introduced in the following chapter. The purpose of this representational reformulation is to provide the language and the organization needed to demonstrate that a regression based parameter estimation algorithm is feasible. The next sections provide evidence that the algorithm is accurate and reliable enough to be worth including as part of a computational scientific discovery system.

3.2.2. Key assumptions and the method to identify parameters

Each equation in the differential equation format of a process model contains three distinct elements. There is the derivative, one or more process expressions, and a coefficient or parameter for each process expression. Both the derivatives and the process expressions can be thought of as vectors that contain a set of values equal in length to the number of measurements. This contrasts with the coefficients that each only have a single value. Derivatives are equivalent to the rate of change of a variable and can be estimated directly from the data. The process expressions are algebraic functions that can be calculated. The parameters are unknown and must be fit according to the available data.

In order to solve the regression problem, both the derivatives and the process expression values must be known at every time step. This can be accomplished by making two simplifying assumptions. The first is that the process expression is explicitly defined as part of the model structure. The second is that the expressions contain only observed variables and known parameters. This assumption allows for the process rate values to be calculated at any time step. Relaxing the restriction that all variables be observed would require the development of a new technique. Fortunately, the requirement for known parameters can be relaxed with the use of a non-linear regression algorithm.

There is also an implicit assumption that it is possible to estimate the derivatives from the data. The estimates may be poor in cases where the data contains discontinuities, is sparse, or extremely noisy, but the algorithm will still work. One additional implicit assumption is that there be more data points than unknown coefficients—i.e. $n > p$. In the example from Figure 4 each equation contains two unknown parameters which requires there be at minimum two data points so that the values can be uniquely identified.

Under those assumptions it becomes straightforward to translate each element of the equations into a format that can be fed into a linear regression algorithm. The elements of each equation include the process expression values, the derivative values, and the unknown coefficients. The regression algorithm returns a set of coefficients that satisfies the relationship defined by each equation. It is clear from this formulation that multiple linear regression can be used to estimate parameters, so long as

there are more data points than parameters. The question remains as to whether the parameters it estimates are the same as the ones that were used to create the data in the first place. This algorithm predicts estimated derivatives derived from observed values. This is in contrast to the previous algorithm that used simulation to predict observed values directly. The expectation is that the regression algorithm will produce coefficients that are a reasonable approximation of the known original parameter values and that the fit to the derivatives will be only slightly different from the original values. Any deviation from the expected parameter values would be due to the error introduced by approximating derivative values empirically.

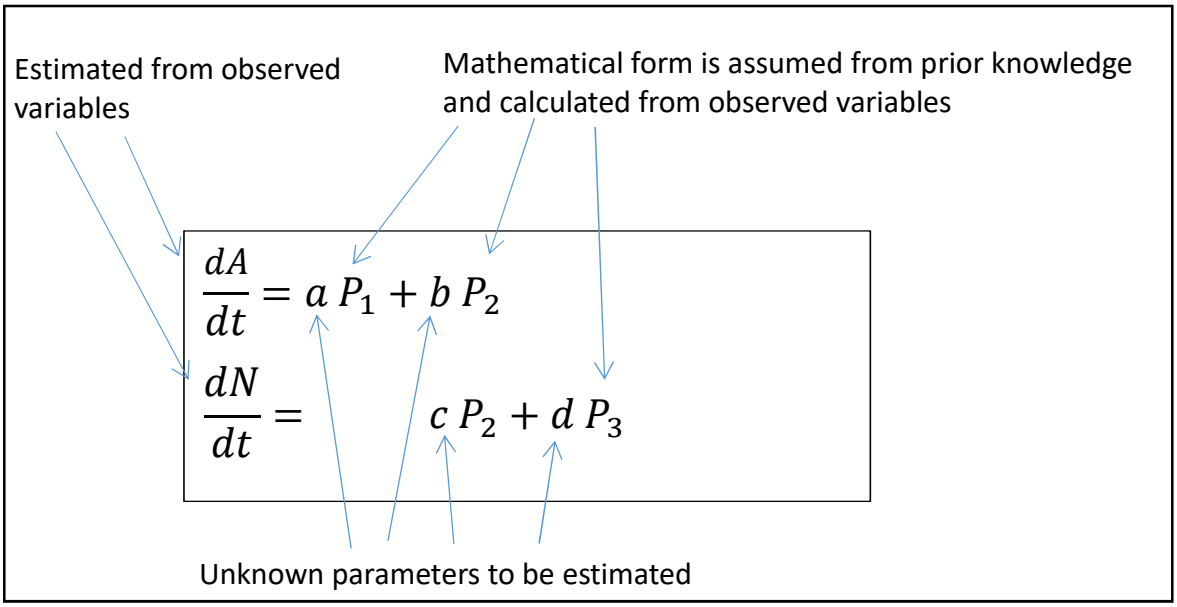


Figure 4 Equation format of a process model formulated to be solved using regression.

The linear regression algorithm chosen for this set of experiments is the *LINEST* function built in to Microsoft Excel 2013. This implementation was chosen primarily due to ease of access and availability. Using an existing implementation will allow this regression-based approach to identifying the parameters of a process model to be tested without being concerned about errors in the regression algorithm. The *LINEST* function accepts as arguments a 1 by *N* vector of values for the dependent variable to be predicted, with N being the number of observations, along with a *M* by *N* array of independent values that are used as predictors where *M* is the number of predictors. The function is also set up to not include a constant intercept term. The *LINEST* function uses a least squares estimation technique to return a set of coefficients that best fit the given data along with an r^2 fitting statistic.

3.3. Sources of error

This section explores the sources of error that were present during the regression algorithm testing. The primary source of error comes from estimation of the derivatives. This will cause the parameters estimated by the regression algorithm to reflect that error. It means that even with noise free synthetic data with a known target model, the parameters that are found by regression will differ from the parameters used to create the data. These parameters are used as part of a differential equation numerical approximation method to predict the trajectories. The numerical approximation introduces a

separate error that could compound the error contained within the parameters themselves. Different numerical approximation methods can also introduce different amounts of error.

The model parameters are estimated based on data derived from the observations and the error minimization technique used to fit the parameters is also based on derived data, not the observed data. The derived data is used as a surrogate to fit the model, however the actual target data is the observed data. Translating between the two data forms in either direction, from trajectories to derivatives using estimation or from the differential equation model to trajectories using simulation, introduces error. The algorithm operates by estimating derivative values of the observed data then using those derivatives along with given process expressions to estimate parameters. It is an open question as to whether the correspondence between regression-based parameter estimates and the observed data is reasonable. This question can be addressed empirically by taking the parameters found by regression then using them to run a simulation and see how closely they reproduce the original data.

3.3.1. Derivative estimation error

The regression algorithm operates on the derivative values so one of the vital aspects of this approach is estimating the derivative of the observed data. The estimation process introduces error in several different ways. The first is estimation error due to discrete time steps. The second is amplification of noise within the data. Noisy data is an important topic that is addressed with more detail in sections 3.5 and 4.5.4. The third is bias introduced by the chosen estimation method.

Estimating a derivative involves calculating the change in the value of a variable divided by the time period over which that change transpires. An estimated derivative will generally be more accurate when a shorter time interval is chosen. The derivative value is usually continuously changing and the estimation makes an approximation that the derivative remains constant over the chosen time interval. Choosing a smaller time interval will minimize the deviation from the assumption of a constant derivative so in most cases consecutive measurements will be used for this purpose in order to reduce estimation error.

The most straightforward estimation method involves calculating either the forward or backwards difference. This means that the estimate of the derivative at the current data point is the difference in the current value and the following value divided by the amount of time between those two values. A forward or backwards difference method is very simple to implement but introduces bias to the data. The result of using a forward difference method is shown in Figure 5. Forward derivatives were used for regression and the algorithm reported r^2 fits of about 0.80. However when the regression parameters were used in simulation it is clear, especially in Aurelia, that there is an overall decreasing trend in the data. Using the parameters found in regression in simulation to predict trajectories is discussed in more detail in 3.3.2. Using backwards derivatives produced the opposite effect with an overall positive trend. The method adopted for this algorithm involves using the centre differencing method that assigns the derivative value of a given data point to be the average of forward and backward derivatives.

One of the issues with estimating derivatives is dealing with the end points as the end point does not have a next data point for forward estimation, and the first data point does not have a previous point for backwards estimation. The method chosen to deal with this problem is to truncate the derivatives and not include an estimate for the end points. This results in the set of estimated derivatives containing two fewer data points than the observed data. There must be more data points than

parameters in order for the system to be uniquely identifiable so dropping two data points could be problematic. Fortunately, time series data of dynamic system that shows complex behaviour will necessarily require many observations to measure those dynamics and systems with simpler behaviour can have additional data points interpolated.

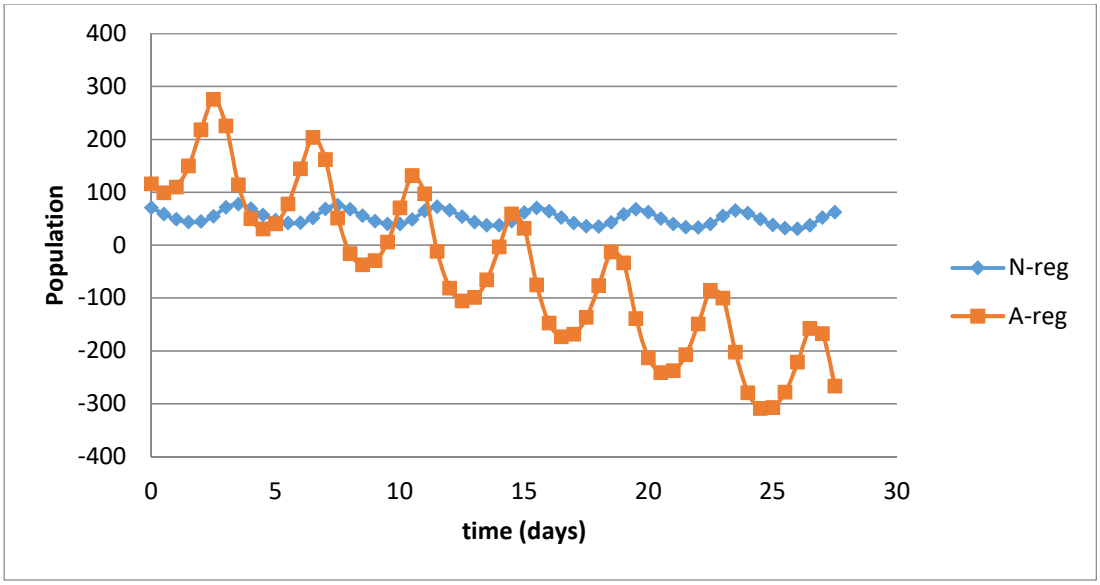


Figure 5 A demonstration of the effects of bias due to using a forward estimation

3.3.2. Simulation

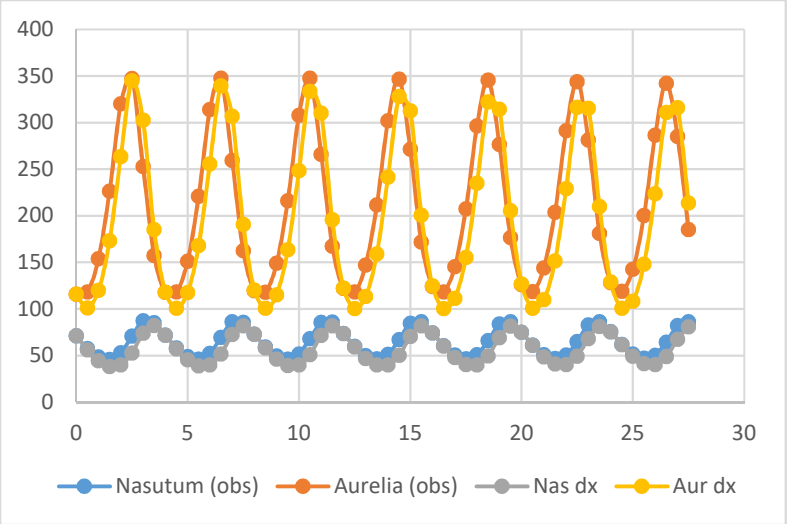
A simulation method is required to perform the next stage of the algorithm testing. Figure 4 shows the example differential equations that are used for simulation. Differential equation models need boundary conditions in order to be solved uniquely. The initial values of the observed variables of provide the necessary boundary conditions for the differential equations from the example. This information allows all the values on the right hand side of the equations to be calculated which then defines the values of derivatives on the left hand side of the equation. Once the derivative is calculated it can be used to approximate future values. One of the simplest approximation techniques is the Euler method. The value of an observed variable at some later time can be calculated using the current value plus the current derivative multiplied by a chosen time step size. The initial values are used to calculate the derivative at the initial time and then the initial value of the variables and derivative values are used to calculate the values of the variables at the next time step. Those new variable values are then used to calculate a new derivative and those two pieces are used to calculate the next data point. This process iterates until the time period of interest is captured.

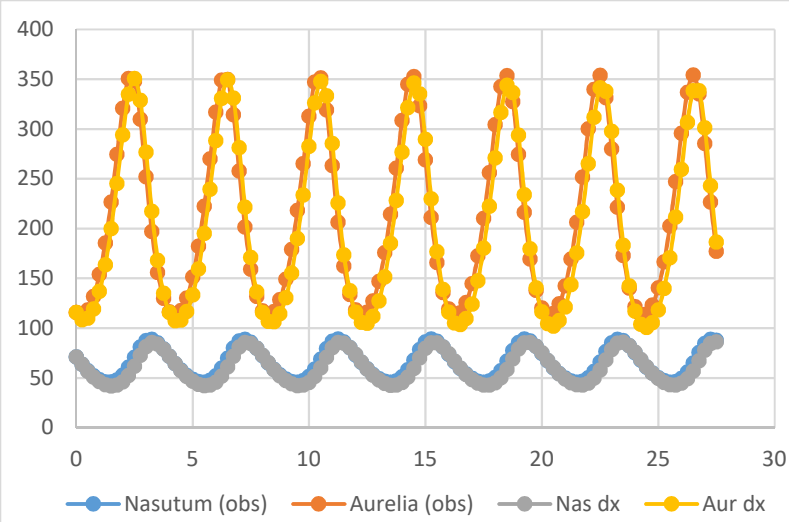
The use of a discrete time step size introduces an error because, in many cases, the derivative value is continuously changing and the discrete time step is only an approximation of a continuous function. The derivative value that is calculated using the model is accurate only for that instant so using the instantaneous value to calculate future values will unavoidably introduce a numerical approximation error. The only way to completely solve this issue is to provide a closed form analytical solution to the set of differential equations. That will turn the differential equations into a well-defined function of time that works for any arbitrary value of time and has no need to iterate or approximate values. Often

it is impractical or impossible to derive a closed form solution to a differential equation so approximations are necessary. One simple way to reduce the approximation error is to use a smaller time step size. The approximation error will approach zero as the limit of the step size also approaches zero. The trade-off of a smaller step size is that more calculations are necessary to produce output over a specified time period. For example if one wanted to know the behaviour of the system 20 minutes from the initial point, using a larger time step of one minute would require only 20 calculations whereas a smaller time step of one second would require 1200 calculations.

Another way to reduce the approximation error is to use a more sophisticated estimation algorithm than the Euler method. Part of the reasoning behind using the Euler method at this stage of the algorithm evaluation rather than one of the more sophisticated simulation algorithms with a smaller error was for ease of implementation. The Euler approach is straightforward and does not require any intermediate calculations so there are fewer places to make a mistake in the implementation. In this phase of the algorithm testing the system is relatively simple with few data points so decreasing the time step is a reasonable way to achieve the desired accuracy.

Figure 6 shows the output of a regression run and two corresponding simulation runs with different step sizes and the calculated sum of squares error (SSE). The graphs provide evidence that the simulation error using a simple Euler simulation method with a time step of 0.5 seconds is relatively small, and that the error when cutting the time step in half to 0.25 seconds is even smaller. It supports the claim that the amount of simulation error is decreased by using a smaller time step. This evidence makes it clear that the parameters found by regression can reproduce the trajectories of the data. The parameters found by regression, including in the table with Figure 6, are all around 10% different from the parameters used to produce the data. The resulting trajectories show that this error is not as apparent in the simulation outputs, especially when using a smaller time step. Figure 6 is evidence that when the parameters found by regression, based on fitting to derivatives, are used to simulate trajectories of the variables the fit is very good. This evidence supports the inference that parameters that fit well during the regression estimation phase will work well during simulation as well. The r^2 score supplied by the regression algorithm on the derivatives is a reasonable proxy for the fit of actual observed data. This eliminates the need to focus on including simulation capabilities as part of the regression algorithm. Simulation still plays an important role in the overall prototyping process because it is needed to create the synthetic data sets that will be used to validate various aspects of a complete scientific discovery system.





Name	True	Estimate	%diff
a	2.715381	2.340645	-13.8
b	-0.04183	-0.03602	-13.9
c	0.004417	0.003881	-12.1
d	-0.93127	-0.81955	-12.0
	Nas r^2	0.992592	
	Aur r^2	0.987707	

SSE Δt=0.5	
Nasutum	Aurelia
4,852	75,369

SSE Δt=0.25	
Nasutum	Aurelia
2,285	42,197

Figure 6 Graphical representations of the difference between model trajectories when using the actual known parameter values and the parameter values estimated from the regression based algorithm. Calculated total sum of squares error at time step 0.5 and 0.25 are also shown.

3.4. Identifiability of processes

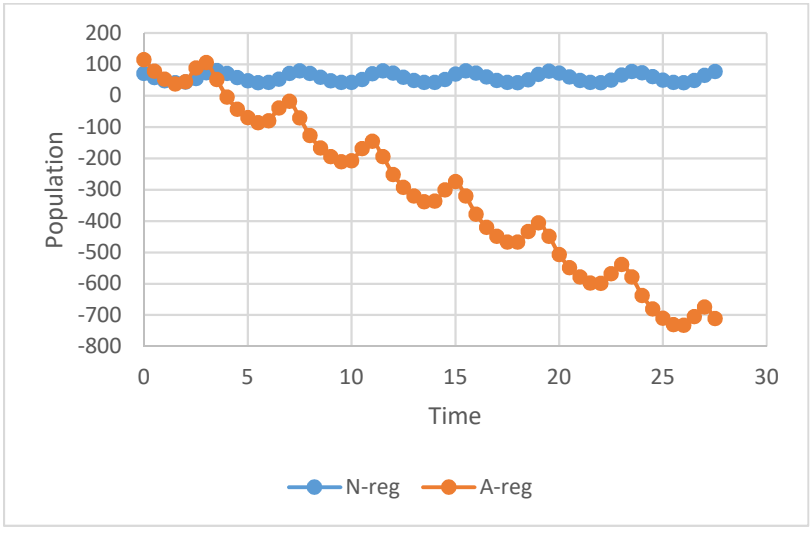
This section explores the ability of the regression-based algorithm to reject processes that do not appear in the model used to create the data. The use of a synthetic data set provides knowledge of the processes and parameters that were used to generate the data. In general, the processes and the corresponding parameters that account for data are unknown and must be found so they are judged based on their fit to the data, not a priori knowledge. The method outlined in section 3.2.2 makes it clear that regression can be used to find parameters. Section 3.3 shows that those parameters that the algorithm found fit the original data well without the need for simulation. It is not obvious what will happen if the algorithm is provided processes that are not identical to the ones used to create the model. The ideal case would be that the algorithm will fail to provide a good fit to the data, thus

rejecting that model structure and providing evidence that the algorithm can differentiate between model structures.

In order to test this idea, a simple set of exploratory experiments have been performed. The process model described in Figure 4 is manipulated so that, in each case, one process expression is changed in some way. Process expressions are allowed to take any algebraic form provided it does not contain unknown parameters. The process expressions are chosen at the discretion of the experimenter and are kept relatively simple to try to capture evidence that equations that have roughly the same number of terms as the target equation can be differentiated by the algorithm. The total number of processes in the model is left unchanged so that the total number of unknown parameters remains constant, four total, in all cases.

The results for each case are presented in an accompanying figure and table that include four components intended to assist the analysis. The r^2 value for each of the two predicted curves, the parameter values found using regression, the original parameters and the percent deviation of the found parameters from the parameters that were used to create the data, and finally a graph of the predicted trajectories of those parameters. The r^2 value is the fitness score reported by the regression analysis. Again, this is measured based on the fit of the parameters to the derivatives which are derived from the observed data. Section 3.3 showed that the correspondence between the derived data and the observed data is acceptable so the r^2 value can be used as a proxy to estimate the fit to the target data. The estimated parameters along with the original parameter values are provided to help understand what kind of values the algorithm can be expected to return. The previous sections have shown that the algorithm works, but it is thus far unknown what happens when the algorithm is presented with a situation where it should fail, and how that failure can be identified.

3.4.1. Experimental results

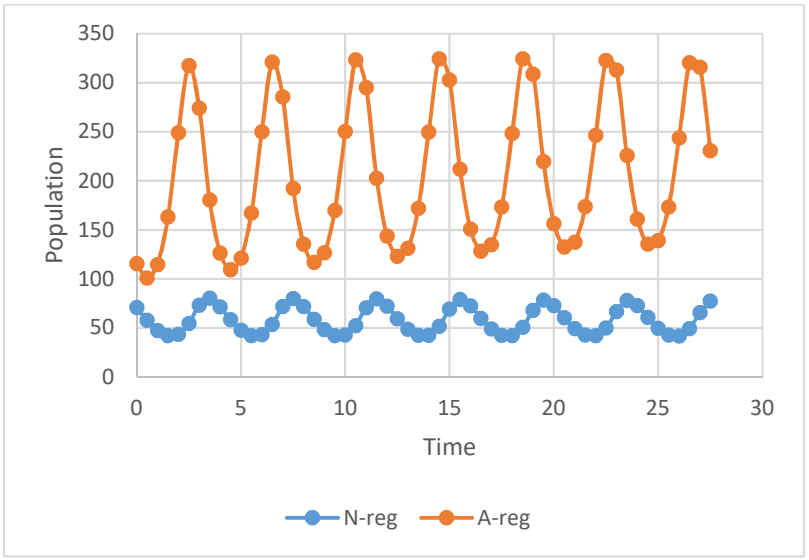


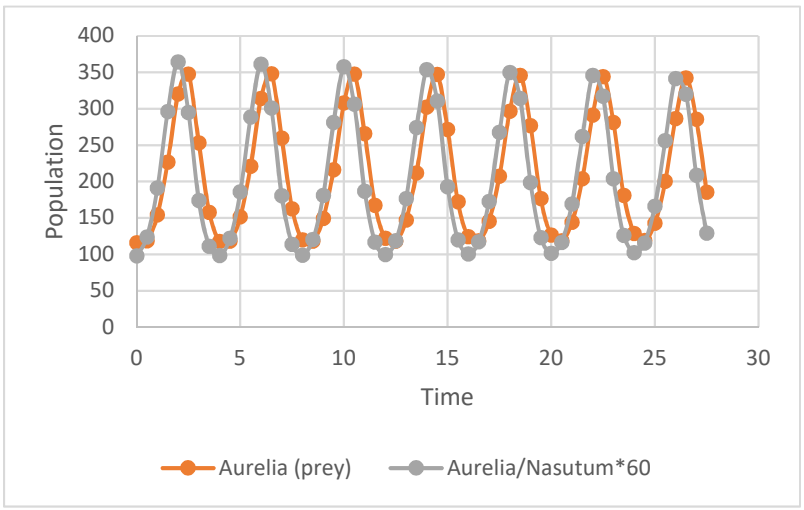
True	Estimate	%diff
2.715381	0.003203	-99.9
-0.04183	-0.01428	-65.9
0.004417	0.003881	-12.1
-0.93127	-0.81955	-12.0
Nas r^2	0.992592	
Aur r^2	0.366802	

Figure 7 The trajectories that result from the best fit parameter values when the wrong equation structure, which uses $P_1 = conc_a^2$, is provided to the algorithm.

Figure 7 shows the results when the expression for P_1 is replaced by an expression that uses the square of the *Aurelia* concentration instead of the original, which was to the first power. The process P_1 only appears in the equation for *Aurelia* so it is not surprising to see that only the fit of that equation is affected, dropping to 0.367. This is an encouraging result because it suggests that using a simple r^2 lower cutoff would be sufficient criteria to reject incorrect processes in an equation.

Figure 8 presents an interesting failure. The expression for P_1 is replaced by *Aurelia/Nasutum* and yet the reported r^2 fits are very good for both equations. This is an unexpected result that can be explained by some subtle mathematical equivalencies. The result suggests that $c_1 * \frac{Aurelia}{Nasutum} \approx c_0 * Aurelia$. This turns out to be true and is shown graphically in the second graph in Figure 8 where the original trajectory of *Aurelia* is graphed simultaneously with the ratio of *Aurelia* over *Nasutum* multiplied by 60. Mathematical equivalencies aside, this process expression fits the data, so there would be no reason to reject it during an actual induction task, unless one places boundaries on the values of the parameters.

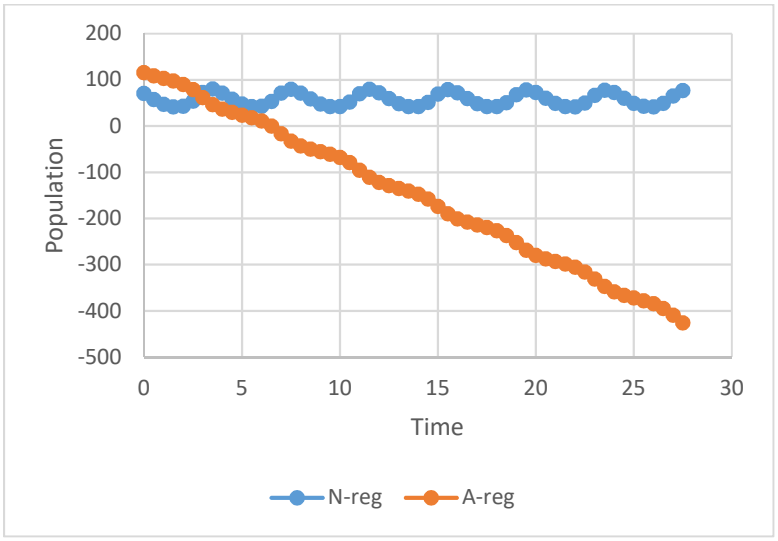




True	Estimate	%diff
2.715381	72.84886	2583
-0.04183	-0.01809	-56.8
0.004417	0.003881	-12.1
-0.93127	-0.81955	-12.0
Nas r^2	0.992592	
Aur r^2	0.987508	

Figure 8 Trajectories resulting from parameter estimation when a model structure that uses $P_1 = conc_a/conc_n$ is given. The second graph depicts the mathematical equivalency between the original data and the ratio of the variables multiplied by a scalar that accounts for the algorithm finding a very good fit.

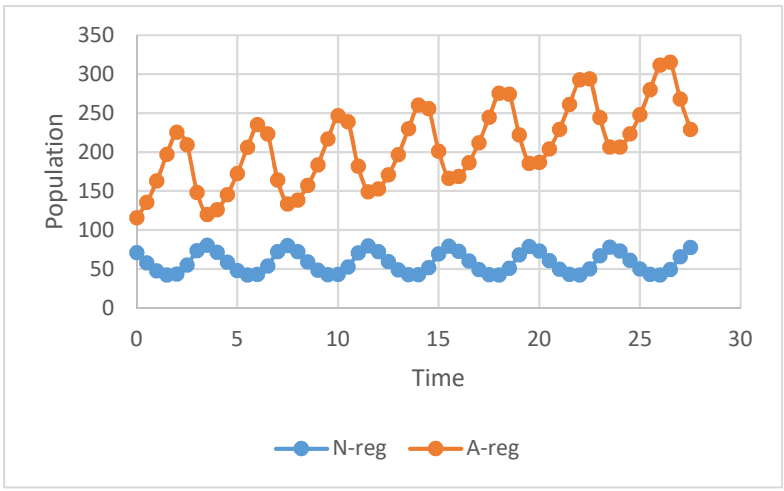
Figure 9 presents the results of substituting the reciprocal of the process expression from Figure 8. In this case the result would be an unambiguous rejection of the process because the fit is very poor, with a reported r^2 value of 0.03. The graph also shows that the behavior of the trajectory of the variable diverges from the target trajectory. Another important idea is that this trajectory quickly beings to predict negative values that are unreasonable because the variable quantity is a measure of absolute population so negative values are unrealistic.



True	Estimate	%diff
2.715381	1.367017	-49.7
-0.04183	-0.009542	-77.2
0.004417	0.003881	-12.1
-0.93127	-0.819548	-12.0
Nas r^2	0.992592	
Aur r^2	0.278	

Figure 9 The trajectories that result from the best fit parameter values when the wrong equation structure, which uses P_1 = $conc_n/conc_a$, is provided to the algorithm.

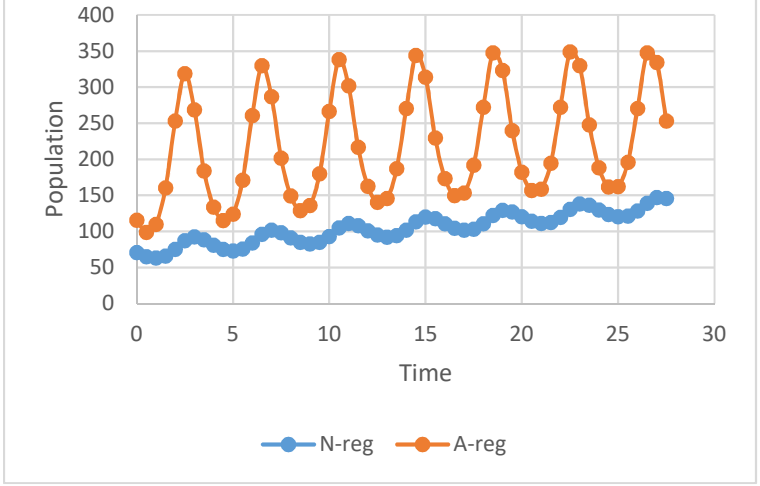
Figure 10 presents the results of substituting the original expression for P_1 with what is known as a logistic growth function. The parameter values used in this equation were taken from some model search results from one of the SC-IPM runs on the empirical data originally used to motivate the synthetic data. The logistic function and its parameters represent a plausible process that could be expected in this domain. Graphically the system displays similar oscillating behavior but the absolute values predicted are incorrect confirming very poor reported fit from regression.



True	Estimate	%diff
2.715381	-4.28961	-258.0
-0.04183	-0.00133	-96.8
0.004417	0.003881	-12.1
-0.93127	-0.81955	-12.0
Nas r^2	0.992592	
Aur r^2	0.033861	

Figure 10 The trajectories that result from the best fit parameter values when the wrong equation structure, which uses P_1 = $conc_a$ (1 - 0.0022 *$conc_a$), is provided to the algorithm.*

Finally, Figure 11 shows the results of a substitution for P_2. In all previous cases the parameters in the equation that was unaffected by the substitution did not change. This process substitution influences both equations and as a result, all parameter values that are estimated are very different. Still, the algorithm produces an unexpected result because fit of one of the equations is very good when they both are expected to be poor. The parameter values used to fit the data are four orders of magnitude larger than the original parameters and so this model could be rejected on that basis if parameter bounds are known. Some of the parameter values also changed signs, which can make them inconsistent with the process definition. In this case P_2 was originally a predation process and the prey species should have a negative coefficient while the predator species has a positive coefficient. Here the signs are flipped, creating a model scenario that suggests the prey species is consuming the predator. There is also an increasing overall trend in the data and it is clear that over time the fit will become worse. Despite the less than ideal result in this case, the results are still encouraging because it seems possible to introduce additional criteria based on the parameter values that can help reject false positives.

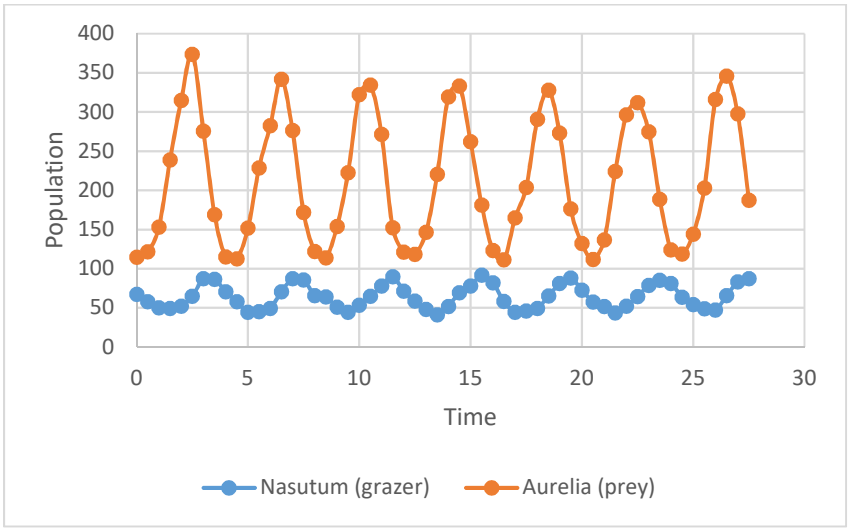


TRUE	Estimate	%diff
2.715381	-2.32695	-185.7
-0.04183	144.2861	-345056.9
0.004417	6.63838	150199.5
-0.93127	-0.31341	-66.3
Nas r^2	0.39898	
Aur r^2	0.9736	

Figure 11 The trajectories that result from the best fit parameter values when the wrong equation structure, which substitutes $P_2 = conc_a/conc_n$, is provided to the algorithm.

3.5. Noise

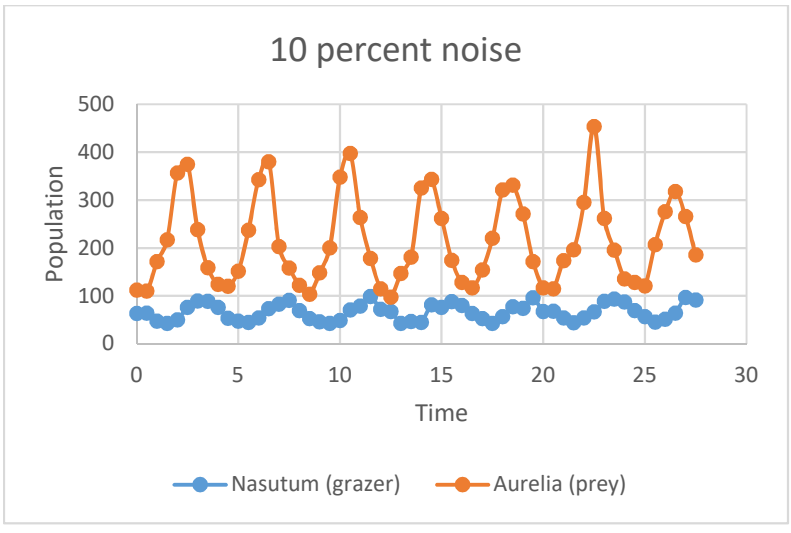
This section provides evidence that the regression algorithm will still perform as intended in the presence of random noise. Section 3.3 talked about how estimating derivatives will introduce error and it is known that noise present in the data will be amplified when taking derivatives (Chartrand, 2011). It is unknown if the noise amplification issue will make the regression-based parameter identification approach ineffective. Empirical testing is used to verify the functionality of the algorithm in the presence of noise. The same synthetic data set used for the previous experiments is subjected to three different levels of noise and the resulting parameter predictions, deviation from the original noise-free values, r^2 fits, are reported along with graphs of the input data.



TRUE	Estimate	%diff
2.715381	2.166539	-20.2
-0.04183	-0.0338	-19.2
0.004417	0.003905	-11.6
-0.93127	-0.82273	-11.7
Nas r^2	0.921277	
Aur r^2	0.935603	

Figure 12 Graphical representation of the model trajectories when five percent noise is added to the data. The table shows the corresponding parameter values from the underlying noise free mode, the best fit parameters returned by the algorithm and the reported r^2 error scores.

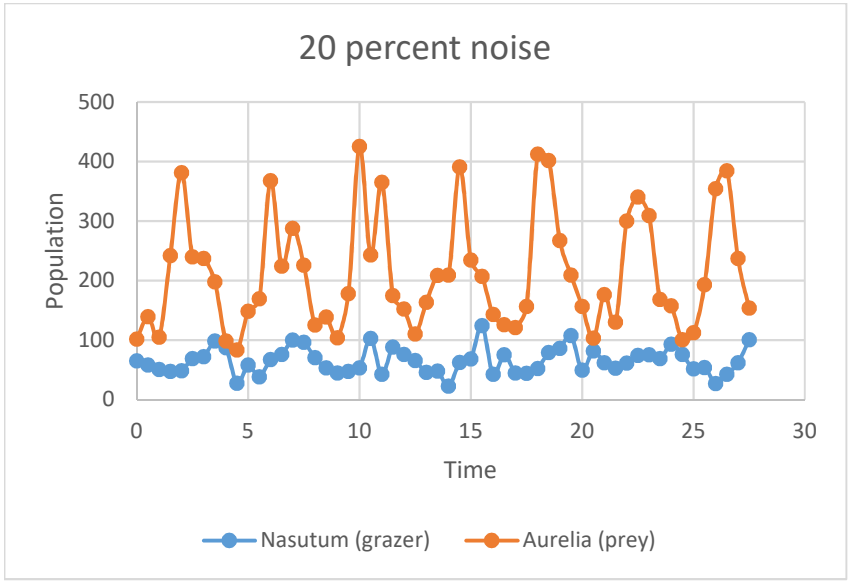
The noise model used in this test added random noise to each data point by constructing a Gaussian or normal distribution with a mean value of 1.0 standard deviation that is a varying percentage of the mean value of the dataset. The first case was five percent, so a random value is drawn from that distribution then multiplied by the first data point. This method is repeated for each data point in the data set until a new noisy data set is created from the original data. Noisy data sets with 5%, 10% and 20% noise were created.



TRUE	Estimate	%diff
2.715381	2.139514	-21.2
-0.04183	-0.03227	-22.8
0.004417	0.003501	-20.7
-0.93127	-0.73793	-20.8
Nas r^2	0.821779	
Aur r^2	0.843668	

Figure 13 Graphical representation of the model trajectories when ten percent noise is added to the data. The table shows the corresponding parameter values from the underlying noise free mode, the best fit parameters returned by the algorithm and the reported r^2 error scores.

The parameter identification method proceeds as before beginning by estimating empirical derivatives for the data. The model structure is already known so the process expressions are calculated using the noisy data and then fed into the regression algorithm to estimate the parameters. Figure 12 shows the data set with five percent noise added to it along with the reported fit by regression. Both curves have a reported coefficient of determination above 0.90 which suggests that five percent noise can be handled by the algorithm with little difficulty. Figure 13 shows a data set with ten percent noise added and the resulting parameters and fit reported through regression. Again both curves have scores about 0.80 which suggests the regression approach may sufficient in some applications. Figure 14 shows the data set with 20% noise added and the resulting parameters and reported fit. In this case it is clear that the regression approach failed with reported fits below 0.4 for both curves.



TRUE	Estimate	%diff
2.715381	0.955913	-64.8
-0.04183	-0.01498	-64.2
0.004417	0.003088	-30.1
-0.93127	-0.65447	-29.7
Nas r^2	0.371944	
Aur r^2	0.280207	

Figure 14 Graphical representation of the model trajectories when twenty percent noise is added to the data. The table shows the corresponding parameter values from the underlying noise free mode, the best fit parameters returned by the algorithm and the reported r^2 error scores.

The results suggest that up to ten percent noise can be handled by the regression algorithm. At some value above this, the available parameters become unable to account most of the variance. It is also important to note that a larger time step in the data will increase the noise amplification effect because instantaneous derivatives would have to be estimated from more distant data points. It is possible that a smoothing method applied to the data before performing regression could be a straight forward way of mitigating the effect of noise if it is assumed that the underlying function is continuous. However, even with no additional data processing it is clear that the regression algorithm will still produce useful results in the face of random noise up to ten percent.

3.6. Concluding remarks

Reformulation of a process model makes it clear that the equations can be solved using regression. Key assumptions are needed that include all variables are observed and that parameter expressions contain

no unknown parameters. This allows regression to be used to identify the parameter values based on estimated derivatives. The resulting parameter estimates are similar to the ones originally used to create the data despite being estimated from a derived data set. The variance between the estimated parameters and the known parameters in the experiments is due in part to error introduced when estimating the derivatives. Error is also introduced during simulation but simulation is not necessary because the coefficient of determination (r^2 score) reported by the regression algorithm serves as a reasonably proxy for actual error. There is evidence that regression approach works when it is given the correct model structure and, just as importantly, it fails when it encounters an incorrect structure which suggests the algorithm can be used to reject model structures. Experiments with noise suggest that the algorithm will be able to handle at least 10% random noise without the need for any changes, in spite of the known effect of noise amplification when estimating derivatives. The concept of regression-based parameter estimation depends only on the principle that the derivatives are linear combinations of observed process expressions. This principle should hold in any domain that can be represented by processes of this form and thus the approach should generalize to those domains as well. All of these factors help support the conclusion that using regression based algorithm for parameter identification as a component in a computational scientific discovery system is a worthy endeavor.

4. Heuristic induction of rate-based process models

Initial investigations into the performance of SC-IPM revealed a fundamental scaling issue. The algorithm relies on repeated simulation to estimate parameters for each model structure. In response, a regression based parameter estimation algorithm was developed. It provides a way to identify parameter values more quickly because it does not require gradient descent coupled with an ordinary differential equation simulation to estimate parameters. In addition, each equation in a model can have its parameter estimated separately. The challenge, then is to implement this idea into a system that can perform these steps automatically to prove the concept and to serve as a computational scientific discovery tool. This requires a new representational framework for process models that uses newly developed rate-based processes that directly predict derivative values. These ideas are incorporated into a Regression-guided Process Modelling (RPM) system that can construct explanatory models of dynamic data. Many of the results and ideas presented in the following chapter are the result of joint work with Pat Langley that has been published as part of the 2015 AAAI conference (Langley & Arvay, 2015).

The work presented in this chapter represents a joint effort between Pat Langley and myself. The initial idea developed in early 2013 following the identification of the problems of the previous system SC-IPM outlined in section 3.1. Pat proposed that the problems could be addressed using a regression guided approach directly on derivatives because it would eliminate the need for simulation during parameter estimation. His initial idea was to assume coefficient values then estimate the process rates then discover the equations for those rates. For example, looking at Figure 4, the initial idea was to make guesses about the values of a, b, c and d, possibly restricted to integer values, then calculate the values of P1 P2 and P3 that fit the observed derivatives. At this stage it was not clear to either of us that a regression-based approach would work due to the data not being independent and identically distributed (IID) and neither of us having experience using regression in this manner. After some time developing data sets to demonstrate the approach I was able to convince him that it was impractical due to there being too many degrees of freedom. There would generally be infinite solutions and no obvious way to choose between them. I proposed that if we knew the process rates it would be straight forward to use regression to estimate the coefficients. His response was it would be easy to calculate process rates so long as only observed variables appeared in the rate equation. These discussions directly lead to the simplifying assumptions that allowed RPM to work.

4.1. Simplifying assumptions for a discovery system

The prototype framework for inducing rate-based process models requires making several simplifying assumptions to make the task of inducing process models more tractable. Some of these overlap with the simplifying assumptions from chapter 3, as the new framework incorporates the regression based parameter estimation algorithm as part of a more comprehensive a discovery system. The first assumption is that the notation must be restricted so that all processes concern changes over time and affect these changes at a specified rate. Chemical reactions provide a good example of a process that has a rate. A single chemical reaction always involves the same inputs and outputs, but the rate of transformation can vary depending on many factors such as temperature or concentrations of the chemical species.

Second, each process influences one or more derivatives that are directly proportional to the specified rate, and that the sign of that proportionality must be defined. For instance, a chemical reaction that combines two substances, C_1 and C_2 to produce a third, C_3 will have derivative expressions of the form $\frac{dC_1}{dt} = k_1 \times R, \frac{dC_2}{dt} = k_2 \times R, \frac{dC_{i3}}{dt} = k_3 \times R$ where R is the rate and each k_i is a constant parameter. In this example, k_1 and k_2 would have to be negative coefficients because they are consumed in the reaction, thus their quantities decrease and so their derivatives would be negative, while k_3 would be positive because this quantity is increased by this chemical process. The equation for each *R* would depend on the specific conditions of the reaction environment. We assume that rate expressions are always positive and would likely be dependent on both C_1 and C_2 as they are required to be non-zero in order for the reaction to occur.

The third assumption is that the rate of each process is determined by a parameter free algebraic expression. In some chemical reactions, this expression is the product of the concentrations of the input substances. Given the concentrations of C_1 and C_2 the rate expression may as simple as $C_1 \times C_2$. The process rates are unobservable so we are free to adopt any measurement scale we like, thus coefficients within the rate expression are not necessary. We recognize that many rate expressions in chemistry can include unknown parameters, such as the Monod expression $\mu = \mu_{max} \frac{x}{(k+x)}$ where μ represents the growth rate, μ_{max} is the maximum growth rate, *x* is the concentration of the limiting substrate and *k* represents the half-velocity constant (Monod, 1949). For now, we exclude these from the framework and intend to address this important issue in future work.

The fourth assumption for this framework is that if a variable appears in a rate expression, it also must appear in a derivative expression associated with that process. For instance, if the process rate is $X_1 \times X_2$ then the process must include derivative expressions that influence both X_1 and X_2, possibly with other variables as well. This rules out processes with terms that influence rates, but are not affected in turn.

We also must make a few important assumptions about the data. Most importantly, all variables within the system must be observed. This is necessary because we use them to calculate the process rate equations. This makes our approach more limited in some ways, but we intend to address these limitations with future developments. Another data assumption is that a variable's behaviour can be represented by a smooth continuous function. The data itself may contain noise but the underlying behaviour is smooth and continuous. This helps mitigate the problem of noise amplification when taking empirical derivatives (Lubansky, Yeow, Leong, Wickramasinghe, & Han, 2006).

4.2. Notation

Table 4 shows several equivalent ways of expressing a rate-based process model that captures the dynamics of a two species predator prey ecosystem. A simple graphical notation of the same model is shown in Figure 15. The variables *x* and *y* are measures of each species population and the system dynamics show a similar oscillating behaviour to the model trajectories of the synthetic data sets shown in the previous chapter. The first notation type in the table is a process centric, the second is a matrix representation, and the third is differential equation. Each notation highlights a different aspect of the same model. The process centric notation provides detail about each individual process. The matrix representation highlights the overall structure of the model showing which equations the processes influence. The differential equations provide a familiar format for simulation of the model. The

graphical notation highlights the structure of the model but leaves off much of the detail about the processes.

The process notation makes it clear that there are three distinct rate-based processes in this model. Each process has a name, a rate, a set of parameters, and a set of one or more derivatives that are influenced by that rate. The name should reflect the purpose of the process and ideally be taken from the taxonomy of the domain in order to help users understand the function of the process. The rate is a parameter-free algebraic expression composed from observed variables. Parameters serve as the proportionality constants that define the effect of that rate on the derivatives of the observed variables. Finally the equations specify which derivatives are influenced by the rate where the notation d[*x*] indicates the first derivative of *x* with respect to time. The process model can be compiled into a set of differential equation model by combining equations.

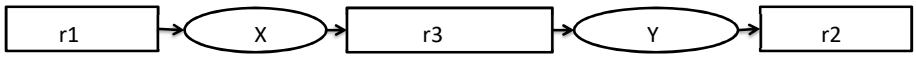


Figure 15. A process diagram for the predator prey ecosystem model in Table 4. Processes are enclosed by rectangles and variables in ellipses. Directed arrows indicate the influence of the process on a variable's derivative.

Table 4. A three process model for a predator prey eco system in process notation, matrix notation, and differential equation notation

```
exponential_change[X]
rate  r1 = X
parameters a = 0.75
equations  d[X]= a · r1

exponential_change[Y]
rate  r2 = Y
parameters b = -0.57
equations  d[Y]= b · r2

holling_predation[X, Y]
rate  r3 = X · Y
parameters c = -0.011
           d = 0.0024
equations  d[X]= c · r3
           d[Y]= d · r3
```

$$\begin{bmatrix} dX/dt \\ dY/dt \end{bmatrix} = \begin{bmatrix} a & 0 & c \\ 0 & b & d \end{bmatrix} \times \begin{bmatrix} r1 \\ r2 \\ r3 \end{bmatrix}$$

$$\frac{dX}{dt} = 0.75 \times r1 + 0 \times r2 - 0.011 \times r3$$

$$\frac{dY}{dt} = 0 \times r1 - 0.57 \times r2 + 0.0024 \times r3$$

The matrix representation from Table 4 can be helpful for understanding the relationships between derivatives and the processes that influence them. Each column of the coefficient matrix represents the influence of a process on a derivative. A coefficient of zero means that the process has no influence on that derivative. The net influence of all processes on a single derivative can be read across a row. Later, when automate model building is automated, matrix notation provides a convenient setup in order to solve the problem of identifying the coefficient matrix.

Differential equation notation is a familiar way to express dynamic systems. This notation is straightforward to input into a differential equation solver. Trajectories can be calculated once initial conditions are provided. It can be more difficult for users to recognize the role that each process plays in the system and thus understand the explanation of the dynamics of the system. This is especially true if terms with zero coefficients are left unwritten.

The graphical notation in Figure 15 shows how the variables, in ellipses, and the processes, in rectangles, connect to each other. The arrows are directed and indicate the influence of the process on a variable. In this example, variable *x* has two arrows connected to it, one pointing in from *r1* and the other pointing out due to *r3*. The arrows indicate that *r1* has a positive influence on the rate of change of *x* and that *r3* has a negative influence. One can also conclude that the derivative of *x* is not directly influenced by any other processes in the model. A variable can potentially be influenced any number of processes so in principle there can be any number of arrows pointing in and out of a variable.

4.3. Regression-guided process modeller

The four simplifying assumptions from section 4.1 state that all processes concern change over time at a specific rate, the influence of a process on a derivative is directly proportional to the rate, the process rate is determined by a parameter free algebraic expression, and that any variable that appears in a rate expression must be influenced by that rate. The four assumptions have been incorporated into RPM, a new system that uses the assumptions to constrain search and make the discovery task tractable (Langley & Arvay, 2015). The system takes as input a set of typed variables, a subset of these variables to be predicted, a set of generic processes of the form shown in Figure 16, and observed time series data for each variable. The program returns a model stated as a set of differential equations, one for each predicted variable, that predicts the derivatives of each variable as a linear combination of process rates. If the system is unable to find acceptable equations for each variable, it returns a partial model containing the equations it was able to find.

The induction proceeds in three main stages. First, the generic process library is instantiated which converts the generic relationships from the domain into specific relationships relevant to the given data. A generic relationship might specify, in a formal way, that a chemical decomposes into two other chemicals. A specific instance of that generic relationship given some data could be that the chemical H_2O decomposes into the chemicals H_2 and O_2. Once all of the possible process instances are created, the second stage requires the system to estimate derivatives then calculate the process rates for each process instance. In the final stage, RPM then iterates through the dependent variables, attempting to find an equation that predicts each one. Processes that are found for one equation can appear again in later equations and RPM employs a greedy search algorithm so potential conflicts are avoided for now.

```
(make-gprocess :name 'holling-predation
               :type        'predation
               :variables   '((x prey) (y predator))
               :dependents  '(x y)
               :influences  '(x y)
               :parameters  '((and (> par 0.0001) (< par 10.0))
                              (and (< par -0.0001) (> par -10.0)))
               :expression  '(* x y)
               :symmetrical nil)
```

Figure 16 An example generic process specification in the RPM system. These are used to generate process instances when definitions of predator and prey variables are provided by the user

Figure 16 shows an example specification, written in LISP, of a generic process in the rate-based process framework used by the RPM system. The first step in the RPM induction algorithm involves instantiating the process instances using the generic process specification, the user-provided type constraints along with the observed variables. Usually the generic process specification will include several different generic processes, each of which is instantiated into several unique but related process instances that are specific to the provided data. The instantiation algorithm begins by selecting the first generic

process in the library and extracting the defined variables and types that are relevant to that generic process. The next step involves examining the provided data to determine which observed variables share the same type. In the case where all observed variables can be any variable type, then all variables will be used to instantiate the generic process which will result in the maximum number of process instances. This creates a set of relevant observed variables that are permitted to appear in a process instance. The next step is to create all subsets of observed variable combinations that satisfy the generic process definition. For example, if the generic process specifies there be one predator variable and one prey variable, the subsets would be all possible combinations of variables that include one predator and one prey. Each subset of unique variable combinations provides the basis for a single unique process instance. The final steps in the instantiation routine involves populating the rest of the instance definition with the observed variable names and evaluating the rate expression.

The name field will ideally have a descriptive name that helps explain the role of the process. In this example, Holling-predation is a relatively simple formulation of the predation interaction between two species. The type field provides a way to implement structural constraints by restricting the number of types that can influence a single variable. In this example one could impose a restriction that only one predation type can appear for a variable, meaning that any prey species has at most one predator species. The variables field specifies the generic variables and types that appear in the instances. The dependents field specifies which variables appear in the rate expression. The influences field specifies which derivatives are influenced by the process. The parameters field places boundaries on parameter values, in this example it states the parameters are between -10 and +10 and not zero. The expression field is the algebraic, parameter-free rate equation. The symmetrical field indicates whether the order of the variables in the expression matter. In this example the order doesn't matter, but for an expression like x/y the order would be important.

Once the system is told which variables are prey or predator it can instantiate all ways that involve the predation generic process from Figure 16, which describes an interaction where the predator eats the prey species. For example, given data with three variables with types defining that A, B, and C can each be either a predator or prey, RPM will generate six instances from the example generic process. A(predator) eats B(prey), A (predator) eats C (prey), B(predator) eats A(prey), B(predator) eats C(prey), C(predator) eats A(prey), or C(predator) eats B(prey). The number of process instances can be reduced if information is available to reduce the type possibilities. For example, if it is known that A is a top level predator, thus nothing in this ecosystem preys upon it, it would reduce the total number of instances in this case from six to four. The set of all process instances form the components from which a complete process model is constructed. The search for models involves testing all different possible combinations of process instances for the combinations that can be fit to the data with an acceptable error.

The next step involves estimating the derivatives for each dependent variable at each time step. RPM uses a centre-difference method, which calculates the derivative of variable X at time t as the average of $X_{(t)} - X_{(t-1)}$ and $X_{(t+1)} - X_{(t)}$. In other words, the derivative now is calculated as the average of the forward and backwards derivative. This first and last data points cannot have their derivatives calculated using this formula, those two data points are truncated so that the working data set ends up having two fewer data points than the original data. Process rates are then calculated from the truncated data. Using the truncated data ensures that the derivatives and the process rates all have the same number of data points. We assume that all variables are observed, and that the process rate equations are parameter free, so it is straightforward to calculate the process rate values from the (truncated) observed data

values. This generates a set of process rate trajectories that correspond to the observed variable derivative trajectories.

RPM now begins to construct a complete model, one equation at a time. The derivative value at each time step is estimated as a linear combination of processes so that linear regression can be used to estimate coefficient values. For each variable's derivative *D,* it considers only those process instances *P* that include *D* as a dependent term, as others would not make appropriate predictors. The system begins by examining all equations of the form $D = a \times P$ using linear regression to calculate the parameter *a*. If the r^2 value of an equation exceeds a user-specified threshold, then RPM adds the best candidate to its model and moves on to the next dependent variable. If no equations exceed the r^2 value, RPM then considers pairs of process instances P_1 and P_2 and testing equations of the form $D = a \times P_1 + b \times P_2$, once again using linear regression to determine the coefficients *a* an *b*. If RPM finds an equation that exceeds the user-specified threshold, it will add the highest scoring equation to the model. Otherwise it will proceed to triples of process rates, continuing to increase the number of processes until reaching a user-specified maximum number of processes per equation.

Knowledge about processes helps modulate the search for individual equations in four ways. First, generic processes include a type specification that can be used to place constraints on processes. This idea was carried over from previous process modelling frameworks including the one used by SC-IPM (W. Bridewell & Langley, 2010). Different processes in the same class can relate the same variables but they do so using different rate expressions. RPM only considers models that contain at most one process instance of each type with the same arguments. Suppose we have two generic predation processes that related a predator population *X* with a prey population *Y*, one with rate $X \times Y$ and another with X/Y. In a system with foxes as predators and rabbits as prey, the generic processes would compile into two process instances of type predation – predation1, with rate $foxes \times rabbits$, and predation2 with rate $foxes/rabbits$. RPM treats these two instances as mutually exclusive as they have the same process type and influence the same variables, ensuring that only one of them appears in a model structure. It captures the idea that in a given model predation between two species occurs via a single process.

The second way that RPM modulates the search for individual equations is by including numeric constraints on the parameters that appear in the equations. This concept was also part of the original framework for process models (Langley et al., 2002) and has carried over into this system. Part of the process definition can include constraints on the values of the parameters that are found by linear regression. Each process defines a rate, and the influence that the process exerts on a variables derivative is proportional to the rate. It is also important to note whether the effect is negative or positive. For a predation process, the predator's coefficient would have to be positive while the prey species' coefficient must be negative. The predation process captures a relationship where a member of the prey species population is consumed, thus it has a negative effect on the population of the prey species. This interaction also causes a positive influence on the predator species, capturing the notion that a successful predator may go on to successfully breed. Constraints on parameter values ensure that the effect of the process on derivatives is consistent with their intended purpose. RPM implements this check at equation construction because linear regression using an r^2 error metric can allow two otherwise identical equations that have opposite signs to produce an equal predictive fit. Parameter constraints allow RPM to differentiate between these equations because only one form will have the correct signs.

The third search heuristic enforces the structural definitions of the process instances. The structural definitions arise from the rate-based process modelling framework because each equation in the model can be fit independently of other equations. In previous frameworks, all of the parameters in the model had to be fit simultaneously rather than one equation at a time. Processes can influence multiple variables, meaning that the rate expression from one process can appear in more than one equation. From a matrix perspective it can be thought of as a column, which corresponds to a process, contains more than one non-zero entry. Thus, once a process is included in a structure for one equation, later equations must also contain that process because you cannot remove a column from a matrix without changing prior equations that have already been found. Thus, the set of processes that appear in later equations can be partially defined by earlier equations that have been found. This heuristic should have the greatest impact on the reduction of the size of the search space.

The final technique utilizes information about previous equations to decide which variable to focus on next. Again this technique is feasible because the rate-based process model framework treats each equation in a model as a separate fitting problem. A partially specified equation requires less search to complete than an equation that is totally unknown. For each variable that hasn't appeared in the model yet, RPM calculates the number of processes in which it already appears and selects the variable that already has the most processes defined. Unless the data set can be partitioned into subsets of variables that do not interact, this heuristic further reduces the search needed to find equations. The first equation that is searched for is a blind search that has to test all combinations, but once this equation is found it reduces the amount of search needed for later equations.

Together, these techniques greatly limit the number of model structures that RPM entertains during its discovery efforts. Empirical evidence of the computational savings due to these techniques is presented in section 4.5.5. In addition, using linear regression (with a least squared error optimization approach to identifying parameters) ensures that a global minima will be found. They make the difference between tractable heuristic search and the intractable methods used in pervious systems.

4.4. Theoretical claims

These assumptions lead to the following theoretical claims about this new framework and this implementation.

- RPM should duplicate previous methods abilities to identify the structure of process models and distinguish irrelevant processes from one that can combine to account for the data.
- The system should identify parameters using linear regression, one equation at a time without the need to simulate the entire model, thus it should be more computationally efficient.
- Using linear regression for parameter estimation should avoid the local optima problem
- The heuristic search that uses previously found processes to guide search should let the system scale well to models of high complexity, provided they are sparsely connected in that each variable is influenced by a few processes.

4.5. Evaluation of the Approach

This section provides an evaluation of the approach that gives support to the theoretical claims. First, a theoretical complexity analysis provides an estimate of the growth in computational time as the complexity of the task increases. Next, a test of the system on empirical data demonstrates basic functionality. After that comes an analysis and discussion of the behaviour of the system when

presented with irrelevant processes. Then comes an analysis of RPM's ability to handle noisy data and an empirical test its ability to scale. Finally, RPM is directly compared to the precursor system SC-IPM.

4.5.1. Complexity analysis

The complexity analysis is encouraging even in the worst case because it shows polynomial growth rather than exponential. For each generic process *P*, there is a combinatorial number of possible instances that depends on the number of variables *V* and the maximum number of variables in a process *k*. It is unlikely that each process will affect all variables. The number of process instances can be calculated as:

$$I = P \times \frac{V!}{(V-k)!}. \qquad \text{Equation 1}$$

Equation 1 assumes that there are no type constraints on the variables so that any variable can play any role. It is the permutation equation for the number of ways that *V* variables can be arranged into *k* slots without repetition but accounting for the order. In the process notation there can be a distinction between different variable orderings, for example foxes preying on rabbits is not the same as rabbits preying on foxes. The maximum number of combinations is then multiplied by the number of generic processes *P* to calculate the total number of possible process instances.

When RPM searches for an equation to account for a derivative it only considers processes instances that influence the selected derivative. We need to calculate the number of process instances that influence just that one variable. It is easier to calculate how many instances there will be should our variable of interested be removed by substituting *V-1* into equation 1, and then subtracting that value from the total number of instances:

$$I' = P \times \frac{(V-1)!}{(V-k-1)!} \qquad \text{Equation 2a}$$

$$I'' = I - I' = \frac{P \times V!}{(V-k)!} - \frac{P \times (V-1)!}{(V-k-1)!}. \qquad \text{Equation 2b}$$

Equation 2a calculates the number of process instances that are created from the variables other than the target variable, then equation 2b takes the difference between that and the total number of instances leaving *I''* which is the number of process instances that influence the target variable . *J* represents the maximum number of process instances that can appear in an equation and it allows us to calculate the total number of possible equations to be evaluated during the search for a single equation via:

$$J' = \frac{I''!}{J! \times (I''-J)!}. \qquad \text{Equation 3}$$

Equation 3 represents the number of ways that the available process instances for a single equation *I''* can be combined into *J* positions without regard to order. In this case the order does not matter because process instances are combined using addition to form an equation.

Assuming the maximum number of variables in a process *k = 3* and the maximum number of processes per equation *J = 3.* This assumption allows an estimate of *J'* to be empirically determined by substituting those assumed values back into equation 3 and 2b giving the formula for *J'* that is only a function of *P* and *V.*

$$J' = \frac{\left(\frac{P \times V!}{(V-3)!} - \frac{P \times (V-1)!}{(V-3-1)!}\right)!}{3! \times \left(\left(\frac{P \times V!}{(V-3)!} - \frac{P \times (V-1)!}{(V-3-1)!}\right) - 3\right)!}. \quad \text{Equation 4}$$

Calculating equation 4 while holding the number of generic processes *P* constant and increasing the number of variables *V* estimates the number of process instances that would be created. The results of this calculation are shown on a semi-logarithmic plot in Figure 17(a). It shows the effect of variables on the growth rate by plotting a different curve for each assumed value of *P*, starting at two and increasing to up to ten. A power series trend line is fit to each curve with the r^2 statistic for goodness of fit also reported. The power series trend line with an exponential of approximately 7.4 is reported along with a r^2 statistic exceeding 0.99 for each curve. This means that when the number of generic processes *P* is held constant, the number of process instance combinations grows at approximately the 7.4^{th} power of the number of variables. Figure 17(b) shows a similar analysis where the number of variables is held constant in each curve and processes are increased. Again a power series trend line is fit to the data and shows that when *V* is held constant, an exponent of approximately 3.0 on the growth of *P* is a good fit.

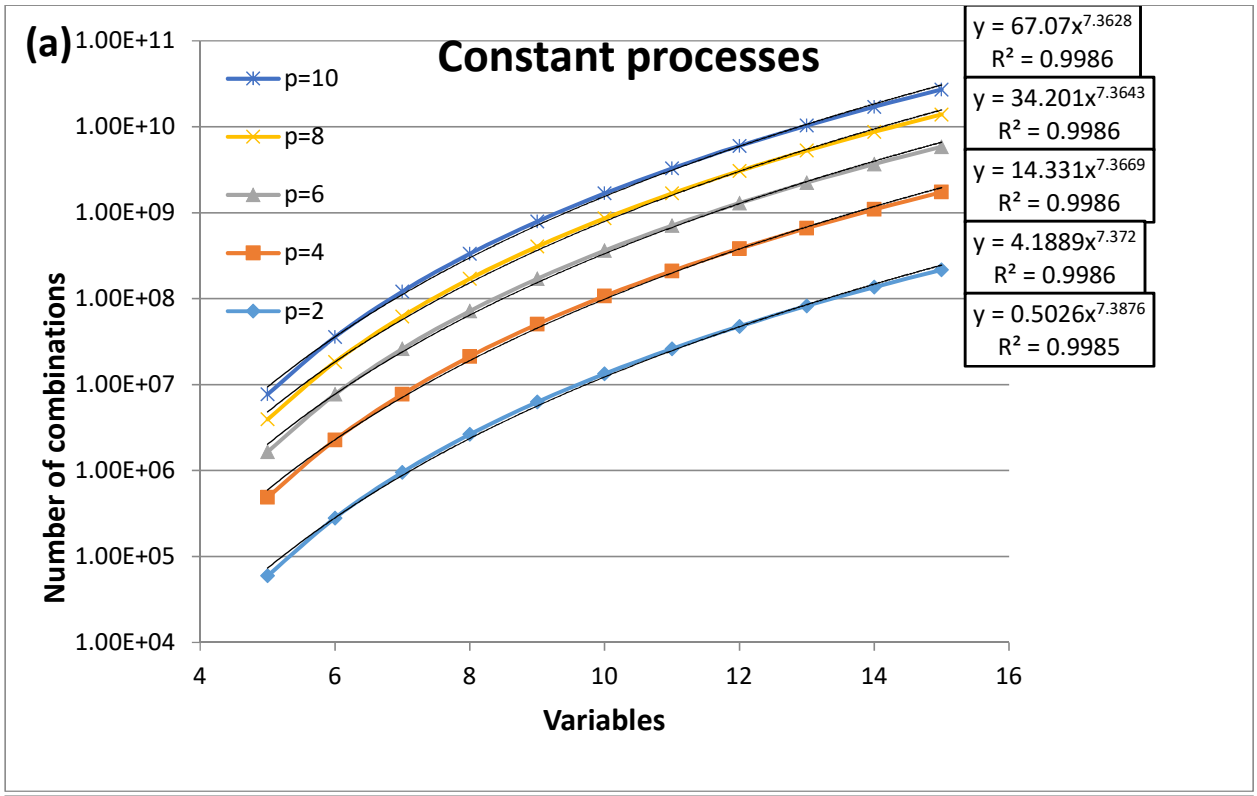


Figure 17 (a) An empirical determination of the growth rate of the total number of process instances when the number of generic processes is held constant and the number of variables is increased. (b) A similar analysis when variables are held constant and processes are increased. Power series trend lines are fit to each curve and the trend line equation and r^2 statistic is reported.

Thus the equation for *J'* which estimates the total number of process instance combinations for a single equation can be empirically approximated by

$$J' \approx P^3 \times V^{7.4} \quad \text{Equation 5}$$

Equation 5 provides an estimate of the time complexity based on the total number of equations. Each equation then requires an evaluation by a linear regression algorithm to estimate parameters. The basic setup for the problem takes the form of equation 6, the symbol × denotes matrix multiplication, not a vector cross product:

$$d\vec{x} = C_1 \times \overrightarrow{P_1} + C_2 \times \overrightarrow{P_2} + \cdots + C_j \times \overrightarrow{P_j}\,, \quad \text{Equation 6}$$

where the vectors *x* and *P* have an entry for every observation. This can be written in matrix format in equation 7,

$$d\vec{x} = \vec{c} \times [P]\,, \quad \text{Equation 7}$$

and with all the vectors expanded in equation 8.

$$[dx_{t=0} \quad dx_{t=1} \ldots dx_{t=n}] = \begin{bmatrix} C_0 & C_1 \ldots C_j \end{bmatrix} \times \begin{bmatrix} P_{0,t=0} & P_{0,t=1} \ldots P_{0,t=n} \\ P_{1,t=0} & P_{1,t=1} \ldots P_{1,t=n} \\ & \vdots \\ P_{j,t=0} & P_{j,t=1} \ldots P_{j,t=n} \end{bmatrix}. \quad \text{Equation 8}$$

Equations 7 and 8 expression matrices where *dx* and *P* contain known values, and the matrix containing the coefficients *c* needs to be solved for. In principle the equation $d\vec{x} = \vec{c} \times [P]$ can be solved for *c* by multiplying both sides of the equation by the inverse of matrix *P,* being careful to respect the order of multiplication because matrix multiplication does not obey the commutative property. It is made a little more complicated because *P* is most likely not going to be a square matrix because the number of observations *N* is usually larger than the number of processes in a single equation. There are various methods to deal with this situation and many of them involve multiplying a non-square matrix by its transpose to create a square matrix and then proceeding from there. RPM uses a matrix inversion-based algorithm to solve for the parameters and the general matrix inversion problem grows by n^3 (Datta, 2010), with *N* being the size of the square matrix and for RPM *N* is the number of observations.

An estimate of the total time complexity with regards to the total number of equations can now be performed. Each time an equation is constructed it must be evaluated against the data. The total time complexity involves knowing the total number of possible equations as well as the size of the data set that it is evaluated against. *J'* from equation 4 estimates the total number of process instance combinations for a single equation i.e. the total number of unique equations for one variable, so multiplying $J' \times V$ gives the maximum number of unique equations for all variables. Evaluating an equation involves performing a linear regression operation that is influenced by the number of data samples *N*.

The time complexity is an estimate of the number of times an equation has to be evaluated by multiplying the maximum number of instances in an equation *J'* by the number of variables, *V* and the number of observations, *N*.

$$total\ time\ complexity = N^3 \times V \times J'. \quad \text{Equation 9}$$

Substituting the empirical estimate for *J'* from equation 5 where it is assumed that *j=k=3* into equation 9 results in an estimate for the overall time complexity:

$$N^3 \times P^3 \times V^{8.4}\,. \qquad \text{Equation 10}$$

Equation 10 shows that the overall time complexity is polynomial with variables being the dominant term. Variables have a much higher exponential term because they contribute to the number of process instances that are created from the generic processes and they increase the number of equations in the model. This analysis ignores the role of heuristics in guiding the selection of variables and the search for equations so we also study the empirical behaviour of our method.

4.5.2. Empirical behaviour on natural data

A simple predator prey ecosystem involving 26 samples (Veilleux, 1979) that captures the interaction between two protozoa – Nasutum and Aurelia – provides a natural data set to demonstrate RPM's induction ability. Earlier methods for inductive process modelling have reported very good r2 scores averaging 0.98 across the two trajectories of this time-series data (Bridewell et al., 2007).

When provided with natural data and two generic processes, one for predation and another for exponential loss/growth, RPM produces the process model shown in Table 4 where Nasutum is variable Y (predator) and Aurelia is variable X (prey). The observed derivative and the predicted derivatives are plotted in Figure 18. The predicted trajectories correspond to an r^2 score of 0.84 for Aurelia and 0.71 for Nasutum. SC-IPM is more accurate in part because it uses simulation to estimate the actual trajectories and does not have to deal with noise amplification. Unfortunately, this comes at a great computational cost.

The space of model structures generated from the generic processes is relatively small but these scores represent the best model that RPM was able to find for this data. There is no objectively correct model, unlike the synthetic data used in the chapter 3 to help design and evaluate the parameter estimation algorithm. In addition to fitting the derivative trajectories well, the process model supports an explanation that Nasutum is the predator with an exponential death rate, and Aurelia is the prey species that grows exponentially. This framework differs from previous systems in part because it predicts derivatives, not the observed variable values. Figure 19 shows a comparison between the observed trajectories and those created by simulating the model discovered by RPM.

The divergence in fits between the derivative predictions in Figure 18 and the observed variable predictions in Figure 19 is due to the introduction and then propagation of error. As discussed in chapter 3, inherent noise in the data is amplified when derivatives are approximated. This error is then propagated during the iterative simulation. Each data point is calculated from previous data points so errors in the model calculations will accumulate meaning that the trajectories will get less accurate over time with respect to the target data. If it is necessary to predict values, it is likely that slightly different parameter values will need to be estimated using repeated simulation. In that case, the parameters used to estimate derivatives can be used as the starting point for a gradient descent method that estimates the variables.

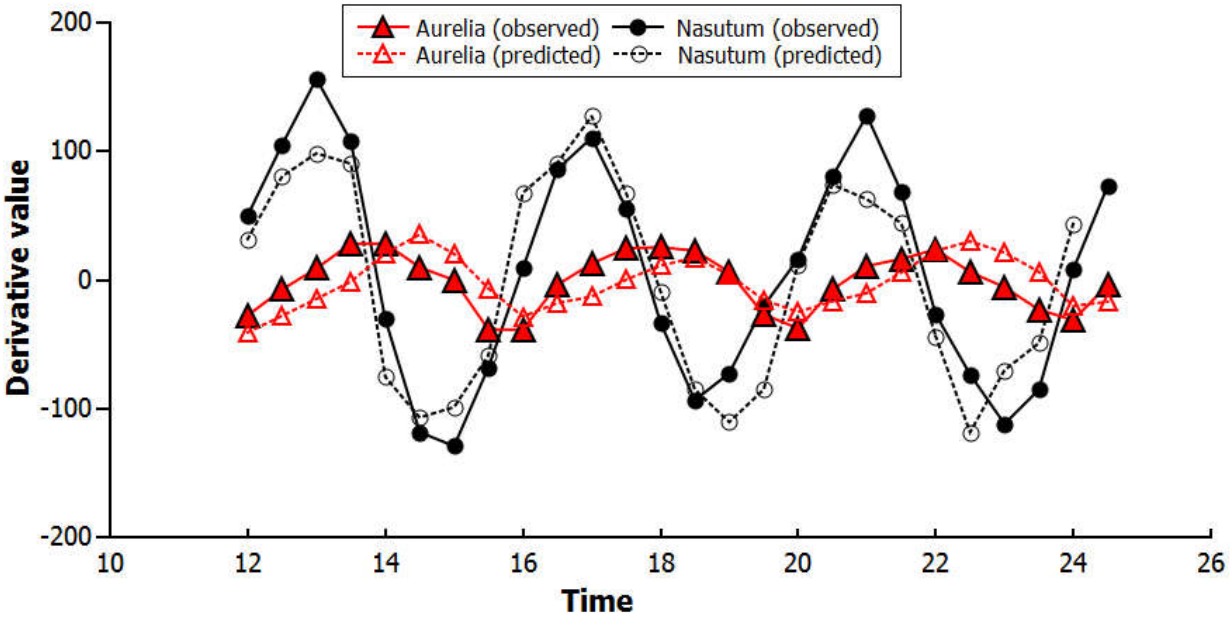


Figure 18 Observed estimated derivatives and model predicted derivative values for a two variable predator-prey ecosystem

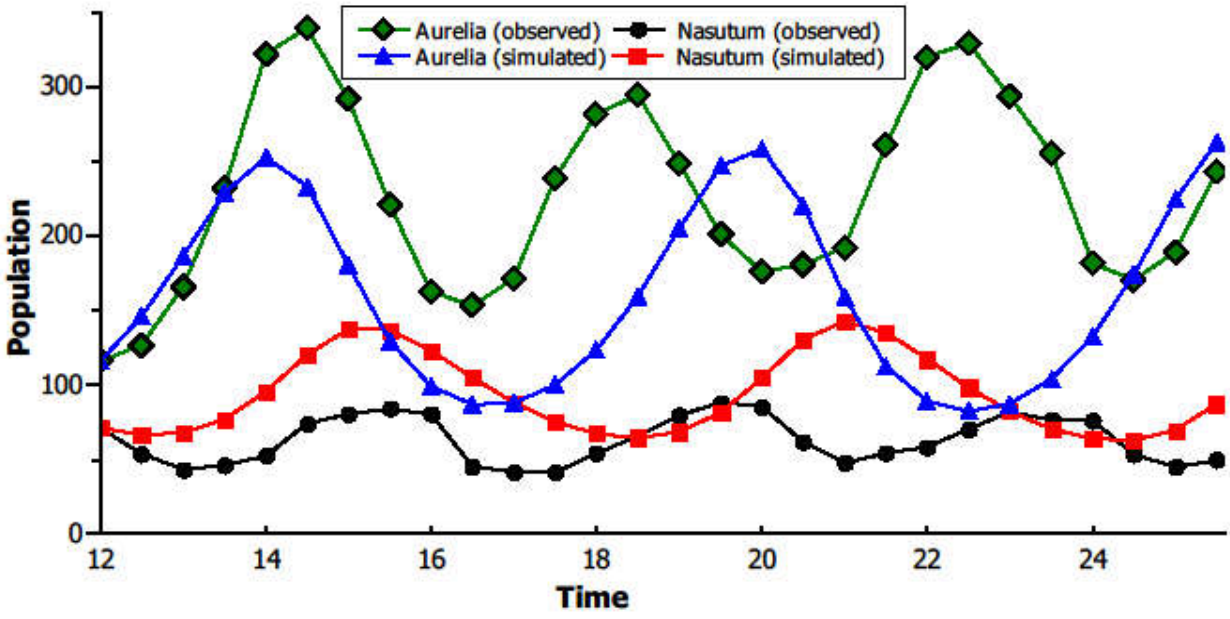


Figure 19 Observed measurements and simulated model trajectories for a two variable predator-prey eco system.

4.5.3. Induction with irrelevant processes

RPM uses regression in order to fit parameters so it is important to verify that regression is being able to differentiate between relevant and irrelevant process instances. The previous chapter showed that the regression based algorithm will reject equations that contain one false process instance, but it is still

unknown if various combinations of irrelevant processes are able to account for the data. It may be possible that the system could find combinations of process instances fit the data thus allowing for any explanation to be as valid as any other. To test this idea, we provided the system with eight additional generic processes for the predator prey domain. These differed from the original ones in the algebraic expression that determines each process rate. Four alternative algebraic forms of the growth/loss process with rate expressions x^2, $x^{0.5}$, log(x), and $\frac{x}{x+1}$ were included. Four alternatives for predation were included with rate expressions of x/y, x^2y, xy^2 and x-y. Results on both the natural data set described earlier, and on synthetic data, show that RPM was able to find the same models as before, suggesting that the system can distinguish between relevant and irrelevant processes.

We are also concerned with the efficiency of induction, so we measured RPM's run time when presented with different numbers of irrelevant processes. More processes increase the size of the structure search space. The red curve in Figure 21 plots the CPU seconds required as a function of the number of generic processes, started with two relevant ones and adding two irrelevant ones on each increment. The growth in computation time appears greater than linear but appears to be slower than our worst case analysis that predicted growth to the third power of the number of generic processes. The difference is likely due to the structural search heuristic, where processes found in one equation are likely to appear in at least one other equation.

4.5.4. Handling noise and complex models

In addition to introducing irrelevant processes, we examined RPM's ability to handle noisy observations. This is particularly important because of the systems reliance on taking empirical derivatives, which can amplify noise (Chartrand, 2011). In order to test this effect, a linear food chain model similar to the model from Table 4 was expanded to five variables. The trajectories for this system with five variables is shown in Figure 20. A linear food chain involves one bottom level organism that is consumed by only one organism above it that is in turn consumed by an organism above it, until a top level predator whose population is controlled through a spontaneous decay rate that approximates the concept of death by natural causes. When RPM is presented with that synthetic data with 5% random noise it returned a partial model that did not match the equations used to generate the data, because it was unable to find an equation for all of the variables.

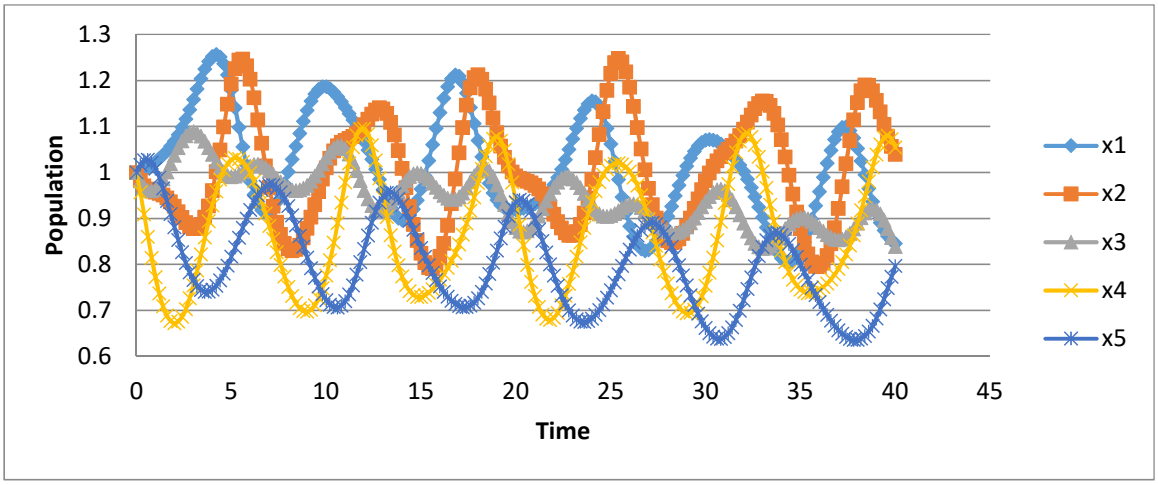


Figure 20 Simulated trajectories of a five variable predator prey linear food chain ecosystem.

The previous chapter suggested that the regression algorithm would be robust to up to 10% noise with little modification. Those experiments were for a single equation and did not account for the role of the search algorithm that uses the results from one equation to constrain the search for later equations, nor did the experiments account for the role of irrelevant processes. RPM uses a greedy search algorithm that helps reduce the size of the search space but also makes the system unable to correct previous mistakes. The presence of noise necessitates reducing the criteria for an acceptable equation which can increase the likelihood that an irrelevant process or a combination of them is able to adequately account for the data. This can result in partial models where some of the equations satisfy the fitting criteria but the system is unable to construct equations to fit for other variables.

Our approach to addressing noise involves applying a locally weighted regression smoothing algorithm (Cleveland, 1979) to the data. This allows the system to find all of the target equations without difficulty. This technique continues to work even when noise level is increased to 10% and we include the irrelevant processes described before. This suggests that pre-processing with simple and inexpensive smoothing techniques will suffice for the approach to handle substantial amounts of observational noise.

We also want to demonstrate that our approach can induce more complex process models that involve many variables. To test this, we ran RPM on a synthetic predator-prey data set with 20 organisms, once again forming a linear food chain. The 20 organism predator-prey model included 236 noise-free data points. The system generated 400 process instances from the variables, but it considered only an average of 1,483 equation structures during search because those found for early variables constrain both which processes to incorporate later, and which variables to search for next. Each equation structure requires a regression analysis to fit parameters and return a fit score. Earlier systems that carry out a non-heuristic search through the structure space would have great difficulty finding models of this complexity. We also systematically changed the number of variables from two to twenty and recorded the CPU time for the system to return a model. Several runs were recorded and in each case the order of the variables was randomized. Randomized variable order was necessary to ensure that the system did not proceed down the same search path in every run. The models used for this experiment are relatively simple, and noise free, so in each run RPM was able to return a model structure identical to the one used to generate the data. In general, the model structure will be unknown. Figure 21, black line, shows the growth in search time as a function of the number of variables. It indicates that the heuristics provide an average case behaviour far better than our worst-case analysis.

Our complexity analysis from section 4.5.1 predicts that computational effort should grow much faster with increasing variables ($V^{8.4}$) compared to increasing generic processes (P^3) and our empirical results depicted in Figure 21 do not seem to support this prediction. It appears that the growth in generic processes curve (red line) is faster than the growth in variables (black line). Part of the difference has to do with the fact that each curve starts with a different number of variables, the red curve is based on a system with five variables. However, that still does not explain why the growth of the “variable” curve is slower than the “process” curve. The discrepancy is likely due to the combination of heuristic search and the sparsely connected model. The heuristic obeys process constraints imposed during search by known equations discovered during earlier phases of the search for a model structure. Processes that are already known are re-used in later equations, effectively reducing the value of k from equation 2a in section 4.5.1, reducing the need for search in later equations. The linear food chain system contains at most two processes per equation so in this particular example k starts at two and if one of the processes

is known *k* reduces to one. In a matrix representation, every row will have exactly two entries because that corresponds to there being two processes per equation, and most of the processes definitions influence two variables, so each process has two values per column. Once the first row is identified, it necessitates that one of the processes that was found to influence this row will also appear in a different row. This structure has at most two processes total so when one is defined the system only has to search for combinations of that known process, and at most one additional process. The effect is greater when more variables are considered because it saves more time with each additional equation, compared to a worst-case analysis that assumes an exhaustive search for each equation without accounting for information about processes to include based on previously discovered equations.

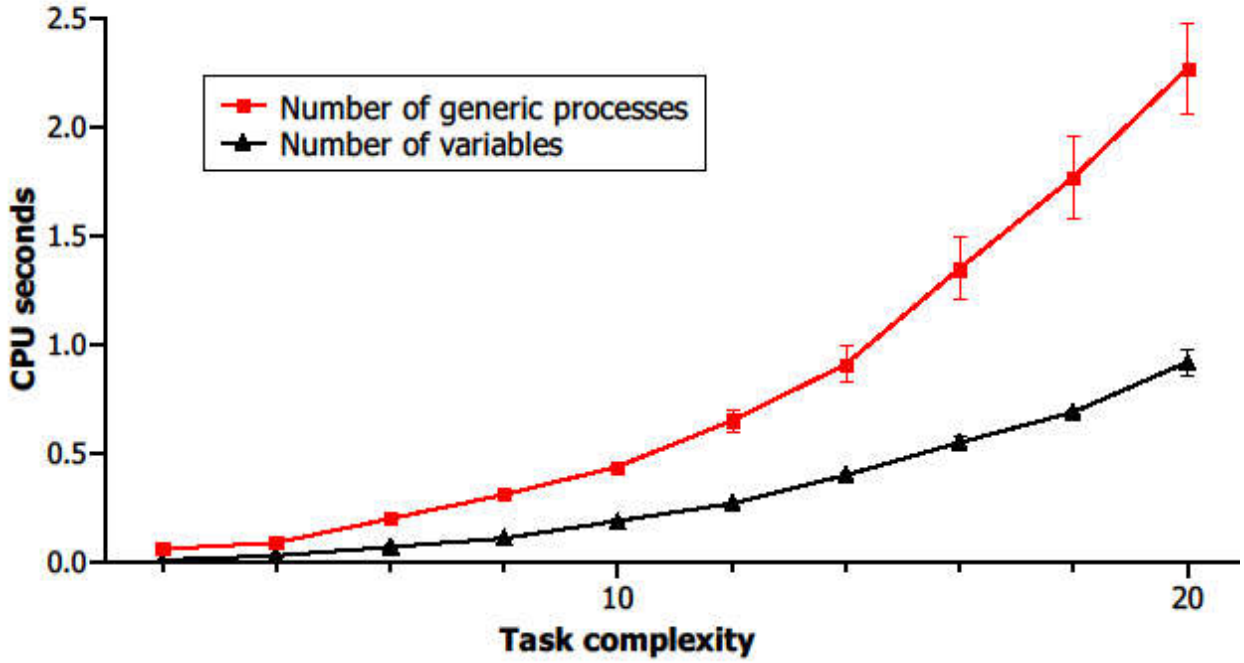


Figure 21 CPU time required for RPM to induce a model for increasingly complex systems.

4.5.5. Comparison to SC-IPM

The final test of RPM is a head to head competition with SC-IPM. This test involves synthetic data with three variables once again in a linear food chain. Both systems were provided with 100 observations of the system and analogous background information.

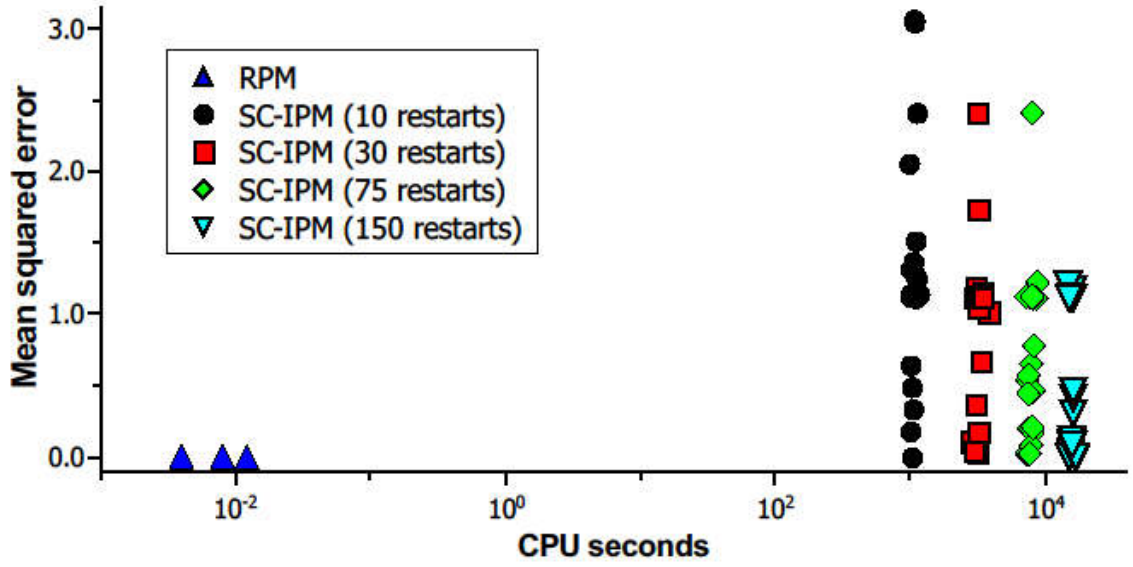


Figure 22 Performance of RPM vs SC-IPM on a model induction task.

SC-IPM has two parameters that RPM does not. The user must specify the number of model structures to consider, and the number of times to restart gradient decent during parameter estimation. The previous chapter showed that higher restarts improve the chances of SC-IPM finding the model. Unlike the SC-IPM experiments in the previous chapter where the model structure was provided, this time we have to tell SC-IPM how many model structures to evaluate. We decided that 600 structures should be sufficient to ensure that SC-IPM examines the target structure. The number of parameter estimation restarts was varied from 10 to 150. At the high end we can expect the parameters to be found around 30% of the time, but a higher number of restarts than this would make CPU times unnecessarily long without further demonstrating the difference in performance.

Figure 22 plots SC-IPM's performance on these dimensions, along with the run time and error for ten RPM runs. The results are unambiguous. RPM is able to find accurate models very quickly and reliably, whereas SC-IPM induces models with comparable accuracy only rarely even on its most expensive runs. Both systems are written in Lisp, but RPM finds these models roughly 83,000 times faster than even the most rapid SC-IPM runs. Each data point on the graph is the result of a model search task, and as expected based on what we learned from the results in the previous chapter, there is a trend towards SC-IPM finding better models as the number of parameter estimation restarts increases. RPM does not report mean squared error so the r^2 score was processed to infer MSE values to make the measurements comparable, keeping in mind that a mean squared error of zero means a perfect fit to the data. The data was synthetic and noise free so perfect fits are possible, although RPM still estimates derivatives which introduces some error even with perfect data.

These experimental results, taken together, offer support for our claim that the new approach induces accurate process models in a computationally tractable manner. The system responds well to irrelevant processes, noisy observations, and substantial model complexity, and it compares very favourably to a representative earlier system.

4.6. Limitations and next steps

The rate-based process modelling framework for computational scientific discovery was developed primarily to address the fundamental scalability limitations of previous approaches. Simplifying assumptions such as processes influencing derivatives, parameter free rate expressions, and all variables being observed provide a framework to implement these ideas. The induction system called RPM introduced in this chapter applies several search heuristics along with a regression based parameter identification algorithm in order to implement the framework into an automated model building system. Experimental evidence shows that when RPM is provided with generic processes and data it can produce a model that explains the observed trajectories. Experimental evidence also suggests that RPM's induction algorithms are more than 80,000 times faster than previous algorithms.

The simplifying assumptions made it possible to produce evidence that the rate-based process model approach can induce process models more quickly than previous approaches. However, the simplifications should be relaxed in future analyses and implementations to increase the generality of the approach. Some of the most important limitations include the greedy search algorithm, the requirement for parameter-free equations, the potential for different processes to have identical rates, and the exhaustive search for individual equations.

The search algorithm limits RPM to finding one model as it takes a greedy approach with no backtracking. As a consequence, once it decides to include a process it cannot later remove it. A better approach could involve a search method that delays decisions about which processes to include until further information is available from other equations. This can also allow the system to return multiple models that satisfy the current r^2 based fitness criteria. Various methods can be used, such as a complexity measure with a simplicity bias, to differentiate between the various models that are returned.

Another problematic issue is the inability of the current system to handle unknown parameter values in the rate equations. Allowing unknown parameters would require changing linear regression for non-linear regression that would then require a gradient descent or similar method to estimate parameters and reintroduce the problem of local optima. However, the approach still works on one equation at a time and non-linear regression would not require simulation of the model so it would likely still be faster than previous approaches. Parameter values that are found for processes in earlier equations will turn unknown parameters into known parameters and later equation searches would not need to re-identify that parameter, further reducing computational effort.

It is possible for different processes to have identical rates. An example of this can occur in chemical domains where the rate expression usually depends on the inputs or reactants but is independent of the products. This can lead to a case where one process states that chemical *X* produces *Y* at rate *X*, or *X* produces *Y* and *Z* at rate *X*. In this circumstance it is possible that RPM will put both processes in the same equation. This will lead to an error in regression because the rate expression *X* appears in two different independent terms. If process type constraints are enforced to address that issue, it can lead RPM's induction algorithm down the wrong path because once it chooses a process it cannot later change its mind and revise previous equations. In the first case, the example process would appear in the derivative equations for *X* and *Y*, whereas in the second case the example process would appear in the derivative equations for *X, Y* and *Z*. Only one can be correct but RPM cannot distinguish which one is wrong unless it is able to analyse multiple equations.

The final issue involves the requirement of an exhaustive search for equations. The algorithm proceeds by testing individual processes, then pairs, then triplets, and so on up to a user specified maximum number of processers per equation in an exhaustive search. This is necessary because at present there is no way to differentiate between candidate process instances. Every process instance that can appear in an equation is equally as likely as any other to be a valid choice to include in an equation. Some method of characterizing process instances could be used to motivate more intelligent decisions about which process instances to try first in the hopes of reducing the number of process instance combinations that need to be evaluated.

Before addressing some of the shortcomings of the system, the next chapter details a critical aspect of the development of a process modelling framework: synthetic data and simulation. Although the key advantage of the regression guided parameter algorithm is that it does not require simulation to estimate parameters, simulation is a critical aspect of creating synthetic data on which to test the system.

The chapter after next expands upon an important advantage of the rate-based process framework. Models are constructed from independent equations that are linked through processes. The difference between the theoretical complexity from section 4.5.1 and the empirical measurements from Figure 21 suggests that knowing some of the processes of an equation ahead of time reduces the amount of search needed to find complete equations. This motivates using partial models as the starting point to induce complete models, similar to the equation discovery system CIPER discussed during the literature review in chapter two. The next chapter describes extensions to the RPM system that allow it to take an existing model and adapt it to new data by either revising the parameter values or finding new model structures when necessary.

5. The creation of synthetic data

The previous chapter introduced the discovery system RPM. It is an implementation of the regression based parameter estimation algorithm that is intended to demonstrate that the algorithm addresses the scaling issue while still maintaining the same discovery capability as previous discovery systems. One of the biggest challenges in developing the RPM system was creating the synthetic data needed to test its performance and capability. This chapter provides details on the development of synthetic data sets. Synthetic data has already played an important role in both the algorithm development as well as its implementation into a complete system.

The regression based parameter estimation algorithm developed in chapter three operates directly on derivatives. That chapter provided evidence that r^2 error is a reasonable metric despite being a measurement of the derivative values, not the observed trajectories. Operating directly on derivatives eliminates the need to perform simulation during parameter estimation, greatly reducing the computational cost of the approach. Despite this, simulation is still a critical aspect of computational scientific discovery using process models, and simulation is a required step during the creation of synthetic data.

5.1. The need and the challenge of synthetic data

Empirical data will be the most common type encountered by any computational discovery system but, during development, synthetic data provides two important benefits over empirical data. First, it can be noise free which reduces the effects of noise amplification when estimating derivatives. This reduces the chance that the model induction system will be led astray by noise, which makes it easier to diagnose and debug any problems with system. Along these lines, the spacing between the data points and the length of the data set can also be controlled. Second, and most importantly, the data is generated using a known model that provides a well-defined target against which performance can be measured. It also provides an unambiguous stopping criteria for the model search. This allows for evaluation to proceed in two major steps, a user creates data from a known model and then the discovery system is tasked with finding that model.

Synthetic data has some important drawbacks that should not be ignored. It is a poor substitute for the empirical data that any discovery system is designed to encounter. The primary purpose of synthetic data is to provide the means to validate and verify the software package in a controlled environment. It should not be used to draw the conclusion that performance on synthetic data will translate to equivalent performance on empirical data.

Creating synthetic data requires performing many of the tasks of model creation that computational scientific discovery is intended to automate. This is one of the main reasons that the creation of synthetic data is such a critical component of developing a computational scientific discovery system. Data from an ecological domain such as the 20 variable models used to evaluate RPM in chapter 4 can be difficult to obtain empirically, so it must be synthesized if systems of that complexity are to be used for testing purposes. Model creation is difficult, especially for complex systems. In some ways, creating synthetic data is more challenging than building a model of known data because the task of creating data does not have a well-defined outcome or target. In addition, research involving processes models makes up a very small portion of the computational scientific discovery community so there are very few examples that use empirical data that can be used to evaluate new systems.

There are a variety of techniques for simulating differential equation models and a large variety of software is available to support this task such as R or MATLAB. The main drawback of using a differential equation simulation engine is that they are not set up to handle differential equations organized into processes. Models are simply defined by equations and it is not straight forward to add or remove processes from a model. This missing feature is the main reason existing software was not utilized to perform the simulation.

There are many empirical data sets across many disciplines that use time series data that is appropriate for process models. Unfortunately, they would not work well in this development application. This work requires the knowledge to create a generic process library that must accompany the data. Many models are expressed in the form of ordinary differential equations and translating a set of arbitrary equations into a process notation is challenging. A more thorough evaluation of the discovery system also requires expanding the process library to include additional processes that may be relevant to the domain but non-applicable to the specific data set. This knowledge is difficult to obtain.

The models used in chapter three for testing during the algorithm development phase were created using an existing model that was developed from empirical data. The example model was simple and sufficient for the purpose of that demonstration. However, the kinds of models needed to demonstrate a discovery system should be more complex than the two-variable three-process model that was used during algorithm development. Ideally there should be data from multiple domains with each data set connecting to a large space of alternative structures. These features of the data can help show the generality of the approach.

The synthetic data and its underlying model needs to satisfy a number of criteria. First, synthetic data requires an underlying process model so that the discovery system has a known target to find. This rules out drawing arbitrary trajectories or finding data that does not have a known model associated with it. The second criteria is that the trajectories should display coupled and complex behaviour. Coupled behaviour makes it unreasonable to assume the trajectories are unrelated and the complexity ensures that they cannot be easily represented by some other simpler modelling method. For example, if the trajectories can be approximated by straight lines one could argue that the variables are unrelated to each other, thus negating the need to explain anything and making prediction trivial. Trajectories can also asymptotically approach an equilibrium value which is non-linear behaviour but also do not make for a very compelling display of model-building ability. One can expect empirical data to vary over time around a relatively steady equilibrium value. Values in nature do not grow without bound and values that decay to zero would probably not be measured. The variance around the equilibrium may be periodic or not, and the oscillations between different entities in the same system may be in phase, or not. A good set of synthetic trajectories should be able to display these types of non-periodic, out of phase oscillations around steady values but do not change amplitude much over time. The data set must also be long enough (in time) to display at least a few oscillations to provide some confidence that the pattern will continue into the future. These synthetic trajectory behaviour criteria are difficult to measure objectively, which, as discussed in the literature review in chapter 2, makes creating process models an ill-structured problem. A third set of challenges involve defining how many variables to include in the model, how to define each process, and how many processes to include in a model. The chosen domain can constrain the process definitions but it is still up to the modeller to choose which processes to include and how many. The purpose of the synthetic data model is to test whether it can be re-discovered, so ideally it should have a plausible structure—similar to the kinds of models we hope

to discover from real data. These basic objectives for data length and behaviour provide enough guidance to begin manually constructing a model.

5.2. Selecting components of a synthetic process model

The first step in model construction is to choose the domain of interest. The model will likely be more plausible if it is intended to be representative of the dynamics from a particular domain. The model induction approach is also organized around domain knowledge that is encoded in a generic process library, so having a specific domain in mind will assist in creating the generic process definitions. Once the domain is chosen, the number of variables in the system can be decided upon. Then the user can begin selecting which processes to include in the system.

There are a few basic guidelines that can be followed when selecting the processes. Constructing a plausible model requires that the modeller either be very familiar with the domain or have an example model to build from. Fortunately the existing model from chapter three provides a starting point to expand upon. That model involved three processes originally shown in Figure 15, replicated in Figure 23 for easy reference. It contained one growth process, one predation process, and one death process. The prey species population grows naturally through its growth process then the population is decreased through predation. Similarly, the predator species grows due to predation, expressing the idea that a successful predator is able to reproduce, and the predator population decreases via a process that captures the concept of death by natural causes. This relationship can be expanded to include more variables by inserting intermediate species that prey upon the species below it to provision its own growth but are in turn consumed by a higher level predator. The relationship is simplified but it is plausible. An expansion of this two variable system to a six variable system is shown in Figure 24.

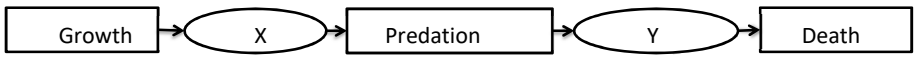


Figure 23. A process diagram for the predator prey ecosystem model in Table 1. Processes are enclosed by rectangles and variables in ellipses. Directed arrows indicate the influence of the process on a variable's derivative.

The processes and parameters that appear in the model are used to calculate derivatives which are in turn used to calculate trajectories from given initial values. The parameter values greatly influence the behaviour of the trajectories so their selection can be a challenge. Domain knowledge can help restrict parameters in terms of values being positive or negative along with upper or lower bounds, but these constraints offer little guidance about the relationship between parameter values and system behaviour. It is helpful for the simulation method to provide fast feedback between changes to parameter values and the effect on trajectories so that the user is able to more quickly try different parameter values.

The final component that the modeller must choose are the initial values of each of the variables. These can also be constrained by domain knowledge. Initial values are generally not as influential as changing parameters or processes but the choice of initial values does affect the behaviour of the system. Once the trajectories are created then any continuous set of data points can be used as the data set, even if they do not contain the chosen initial values from the simulation, and the discovery system would recover the correct processes and parameters. There can be exceptions if initial values are chosen

outside of the characteristic values of the system, for example having very large values. Another possibility is if a value starts at the exact equilibrium of the system, then the values may remain at that equilibrium. The equilibrium values of the system are usually unknown at the time of model creation so some trial and error is sometimes needed during construction.

In summary, the components of a model that produces synthetic data should have a number of features. An underlying process model should be present so that the discovery system has a known target which helps troubleshooting in cases where the target isn't found. The trajectories that the model produces should display complex behaviour that mimics empirical behaviour from the domain. There also must be enough variables in the system with complex interactions so that the system cannot be dismissed as a toy or otherwise unrepresentative of the complexity of real systems.

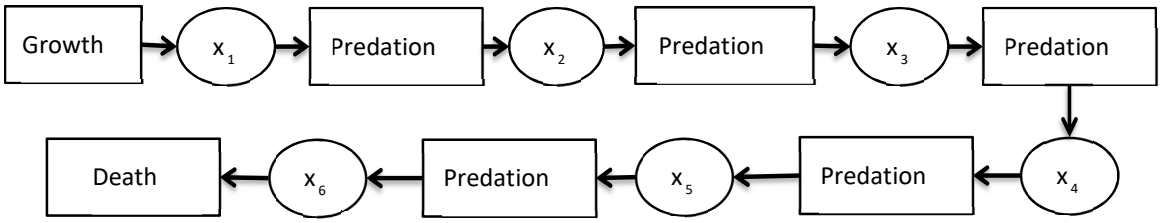


Figure 24 An expanded predator prey ecosystem model constructed by expanding a simpler model found using empirical data

5.3. Simulation algorithm

Once the domain, the processes, the parameter values, and the initial values have been chosen then it is time to simulate the model and see what kind of trajectories the chosen combination of components produce. Figure 25 displays an example matrix representation and the equation representation of a process model that contains three variables and three processes. Performing matrix multiplication produces the equation format. In order to create synthetic data, the modeller must provide both the coefficient matrix and the process rate vector. These will define a set of differential equations that can be used to simulate trajectories.

$$[Derivatives] = \begin{bmatrix} Coefficient \\ Matrix \end{bmatrix} \times \begin{bmatrix} Process\ Rate \\ Vector \end{bmatrix}$$

$$\begin{bmatrix} dX/dt \\ dY/dt \\ dZ/dt \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \times \begin{bmatrix} P \\ Q \\ R \end{bmatrix}$$

$$\frac{dX}{dt} = aP + bQ + cR$$
$$\frac{dY}{dt} = dP + eQ + fR$$
$$\frac{dZ}{dt} = gP + hQ + iR$$

Figure 25 Matrix and differential equation formats of a process model

A differential equation, such as shown in Figure 25, are functions that compute a derivative. Initial values provide an initial input to the function so that the initial derivative can be computed. The initial derivative value and the initial values can then be used to estimate a future value by multiplying the derivative by a length of time and adding that quantity to the initial value. This produces an estimate of new values at some future time. The estimate will be exact if the derivative is constant over the time interval, often a rare situation for complex systems. These future values are then input into the function to compute a future derivative and the procedure can iterate for as long as needed.

Chapter 3 discussed performing simulation using the Euler method which is simple and easy to implement. The error due to simulation was minimized in that chapter by using a smaller time step, but that increased the computational effort and the memory needed to keep track of all the data points. Simulation error can also be reduced by using a more sophisticated simulation algorithm such as the 4th order Runge-Kutta method (Ascher & Petzold, 1998). This method achieves a more accurate result by making four approximations of the future value and averaging them. This particular method was chosen in part because the previous discovery system SC-IPM used this algorithm for its simulation engine so it provides a way to verify that the simulation outputs are correct. The Euler method described in chapter 3 uses the derivative value along with a user specified time step to make an approximation of the future value of the variable. I found that graphical representations were helpful in explaining the simulation algorithm concepts to those who had less experience in working with numerically approximated ODE models. Figure 26 through Figure 30 show the steps using a graphical representation and each step is described in detail in the following paragraphs.

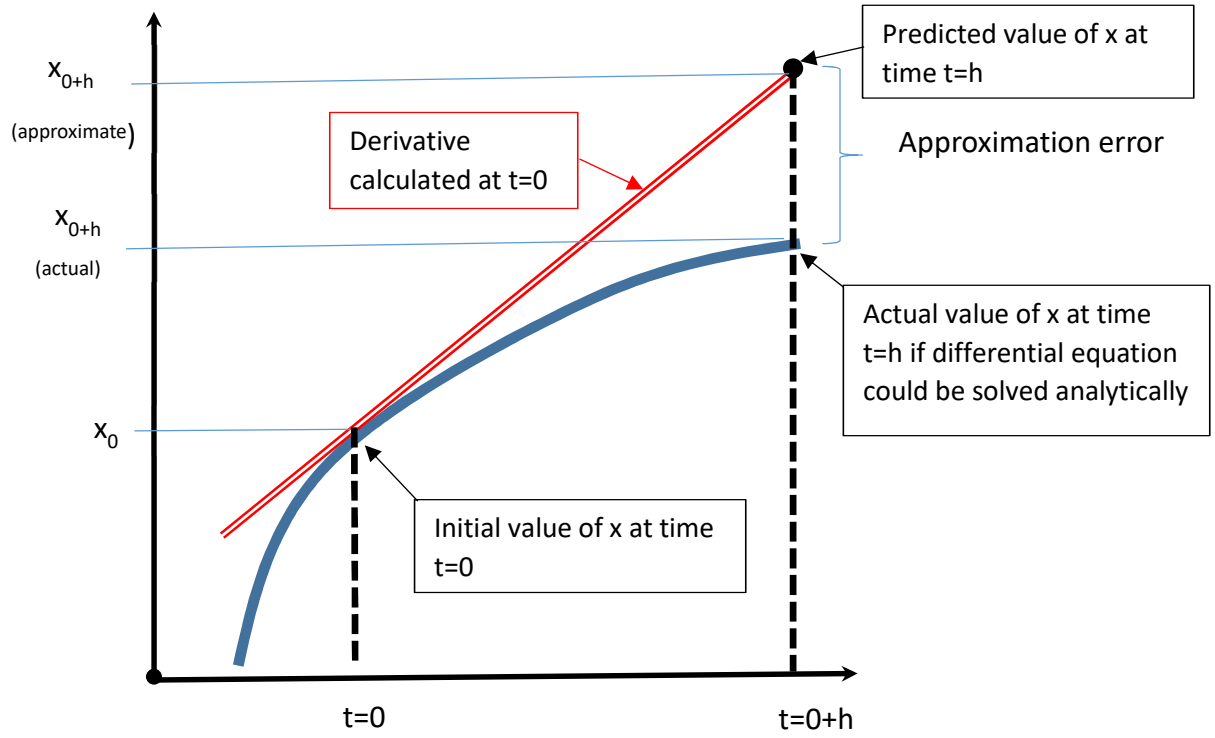


Figure 26 The initial derivative calculation of the 4th order Runge-Kutta method shares similarities with the Euler method, which estimates the functional value x at h time units into the future using a single derivative estimate. See text for further details.

Figure 26 shows a schematic graphical representation of the first step in the RK4 algorithm. The derivative is calculated using the model equations and the initial values at time *t = 0* and the derivative value is represented by the red double line drawn tangent to the solid blue curve. The red double line has a slope equal to the value of the derivative at *t = 0*. The blue curve represents the actual trajectory of the variable *x* which this simulation example is intended to approximate numerically. The first step of the RK4 algorithm is nearly identical to the Euler method, which only uses one derivative calculation to estimate future values. The figure demonstrates the cause of the error inherent in all numerical approximation methods. The derivative value, or the slope of the tangent line, is used to calculate an estimate the value of *x* at a future time *t=0+h* using the formula $x_{0+h} = x_0 + \frac{dx}{dt} * h$ where *h* represents the size of the time step, x_0 is the initial value and *dx/dt* is calculated using the model equations. The RK4 method differs from the Euler method at this stage because it calculates the value of *x* at future time *t = 0+0.5h*, halfway to the desired estimate, then uses this value for future calculations.

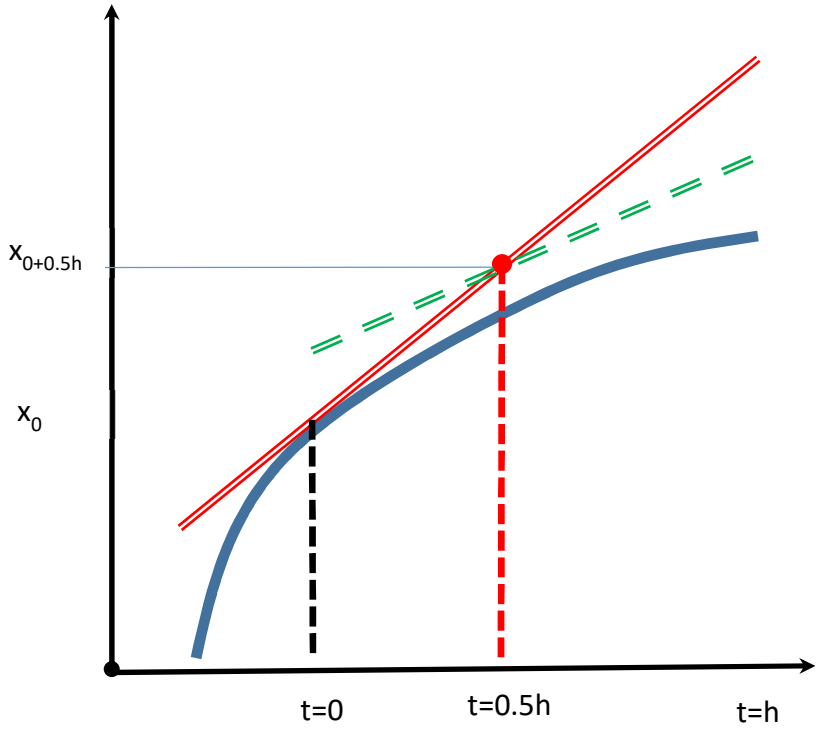


Figure 27 The derivative approximation of the second stage of the RK4 algorithm. See text for more details.

Figure 27 shows a schematic graphical representation of the second stage of the RK4 algorithm. The slope of the first calculated derivative, the double red line, is used to estimate a point half way to the target value using the formula $x_{0+0.5h} = x_0 + \frac{dx}{dt} * 0.5h$. The midpoint value is used to calculate the derivative at the midpoint. This slope is represented graphically by the green dashed double line in the figure. The schematic shows that the derivative being approximated at *t = 0.5h* has error. The blue curve represents the output of the functional form if the differential equations were solved analytically. The purpose of this stage is to try to approximate a tangent line drawn on the blue curve.

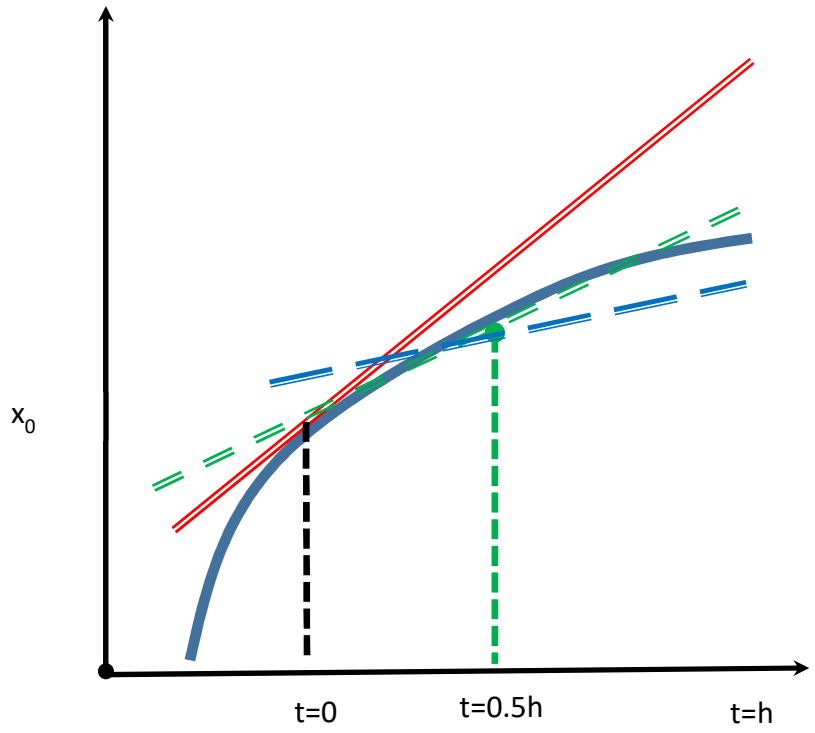


Figure 28 The third derivative approximation in the RK4 algorithm

Figure 28 shows a schematic graphical representation of the third stage of the RK4 algorithm. A similar procedure is repeated to calculate a midpoint value using the same formula $x_{0+0.5h} = x_0 + \frac{dx}{dt} * 0.5h$. The key difference is that in this case the derivative *dx/dt* is the slope of green dashed line instead of the slope of the red double line. The visualization in the figure shows then green dashed line being transposed back to the initial point and then used to estimate a new value at *t = 0.5h*. The next calculated derivative is represented by a dashed blue double line. This second midpoint value is then used to compute yet another derivative value represented by the blue double line with longer dashes.

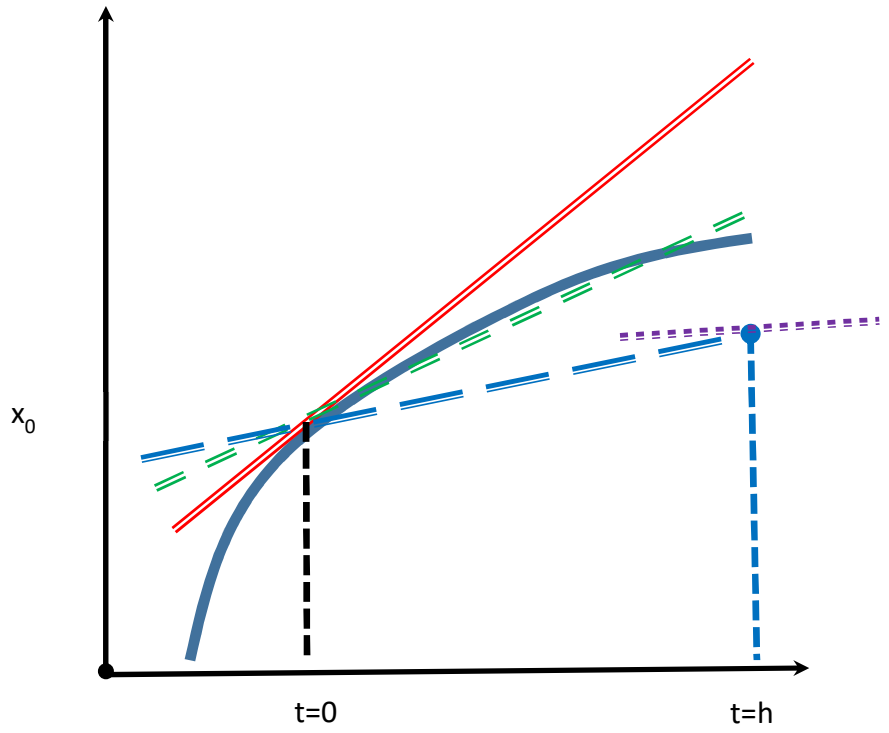


Figure 29 the fourth approximation of the RK4 algorithm taken at t=h

Figure 29 displays a representation of the fourth stage of the RK4 algorithm. It involves repeating a similar pattern as before, using the just calculated derivative represented by the blue dashed line to estimate the value of *x* at the target time *t = h*, not the midpoint. That estimate of *x* is then used to compute the derivative at *t = h*, displayed in the figure as a purple dotted line.

After the fourth stage there are four distinct derivative estimates, written in shorthand as d_1 through d_4. d_1 is the derivative calculated when *t = 0* represented by the red double line. d_2 is calculated using d_1 at the point *t = 0.5,* its value is represented by the green dashed double line. d_3 is calculated using d_2 at the midpoint when *t = 0.5h*. It appears as the blue line with long dashes. Finally, and d_4 is calculated using d_3 when *t = h* and appears as the purple dotted line. The four derivatives are averaged using the following formula: $d_{avg} = \frac{(d_1+2d_2+2d_3+d_4)}{6}$, with the two midpoint estimates being weighted higher. This average slope value is then used to produce the final estimate of the next data point using the same formula as before $x_h = x_0 + d_{avg} * h$. Figure 30 shows the schematic graphical representation of the averaging step where the four slopes are combined using a weighted average to estimate the next data point.

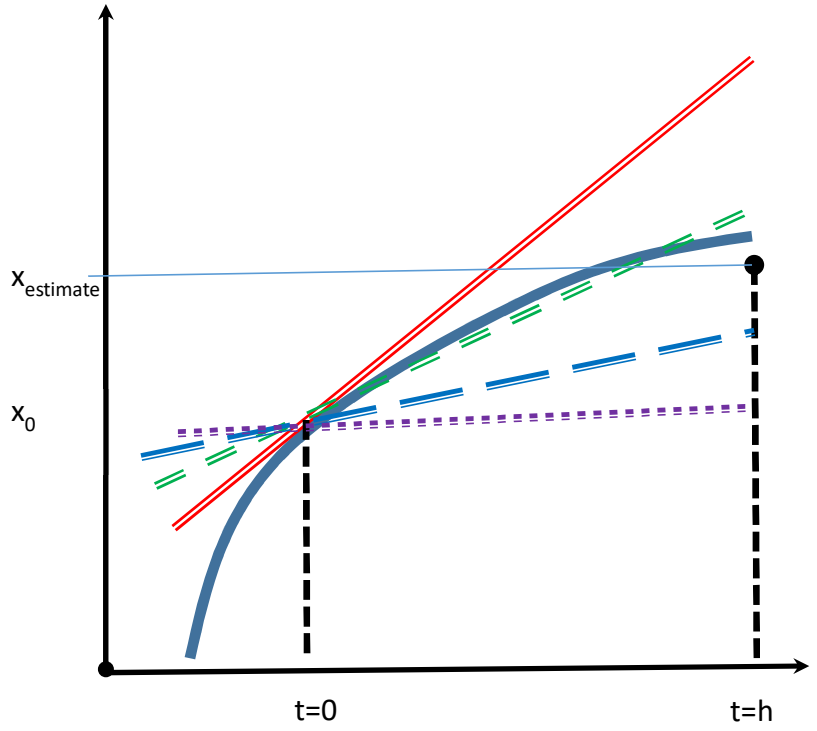


Figure 30 The final approximation is a weighted average of the four approximation with higher weight being given to the midpoint derived derivative estimates.

5.4. The model building method

The data simulation implementation was created using Microsoft Excel and provides several useful features. The modelling approach requires the user to select and keep track of process rate equations, initial values, and parameters. Each of these components can be written out in the spreadsheet in separate cells, making it a convenient way of keeping track of the model components for future reference. The functional forms of the process rate equations can also be written out using an easy-to-read format and labelled for easy reference. There is also a built-in graphing utility that updates trajectories instantaneously as soon as any coefficient values or initial values are changed, which happens frequently during model construction.

The main disadvantage is that the data generated from the spreadsheet-based simulation needs to be reformatted before given to the discovery system. Another issue is the lack of flexibility of the spreadsheet to reference a function rather than a cell value. When the user wants to test a different structure, which usually involves changing a process rate equation, the functional form of the new equation must be updated in 4 different places (one for each RK4 calculation). It is possible to create custom macros that would allow for this capability but that has yet to be done.

5.4.1. A predator prey domain example

Table 5 displays the model components of the expanded linear food chain model shown in Figure 24. The components are displayed as they could appear in a spreadsheet implementation. The model contains 7 processes with the process rate equations included in the last column. The initial values are also defined, appearing below the coefficient matrix. The coefficient matrix can be multiplied by the rate matrix in order to produce the model equations that appear below the table.

Several decisions were made when selecting the components that were included in this model. The chosen domain was the predator-prey ecosystem. There is an existing model in this domain that can be expanded into a more complex structure and all of the processes that appear in this domain are well defined and understood. Six variables were selected for the size of the expanded system because it more than doubles the number of variables from the simple system (described in chapter three during algorithm development) but doesn't have so many variables that the manual calculations become unwieldy in the spreadsheet. Model induction by the discovery system will also be faster with fewer variables, which can help reduce computation times during development. The parameter values were selected with no guidance, so all unknown parameters were set to 1, respecting the relative signs enforced by the process definitions. A similar decision was made for the initial values as no guidance is available for what the values should be.

Table 5 The components of a process model with six variables that has the model structure shown in Figure 24

vars	R1	R2	R3	R4	R5	R6	R7	R1=	x1
dx1	1	-1						R2=	x1*x2
dx2		1	-1					R3=	x2*x3
dx3			1	-1				R4=	x3*x4
dx4				1	-1			R5=	x4*x5
dx5					1	-1		R6=	x5*x6
dx6						1	-1	R7=	x6

Initial values	x1	x2	x3	x4	x5	x6
	1	1	1	1	1	1

Model Equations

$$d[x_1] = 1 \times x_1 - 1 \times x_1 \times x_2$$
$$d[x_2] = 1 \times x_1 \times x_2 - 1 \times x_2 \times x_3$$
$$d[x_3] = 1 \times x_2 \times x_3 - 1 \times x_3 \times x_4$$
$$d[x_4] = 1 \times x_3 \times x_4 - 1 \times x_4 \times x_5$$
$$d[x_5] = 1 \times x_4 \times x_5 - 1 \times x_5 \times x_6$$
$$d[x_6] = 1 \times x_5 \times x_6 - 1 \times x_6$$

The interaction between the variables in this model is based on a predator-prey ecosystem. The term *dx1* in Table 5 represents the time derivative of the variable *x1*. *R1* through *R7* are the processes that appear in the model and their associated rate expressions are written out in the last column of the table. The actual process definitions are written out explicitly in this format but all of the relevant information about processes, except for the name, is available in the table. In this example, *R1* is based on a growth process, *R2* through *R6* capture a predation interaction, and *R7* is a death process. Reading across its row in the table we can see that *dx1* is influenced by *R1* and *R2*. The influence of a process on each derivative can be read in the columns of the coefficient matrix. *R1* has a single entry, a positive 1, for *dx1* and the rest are zero value, represented by an empty field in the table. This means that *R1* influences *dx1* in a positive manner and has no effect on any other derivatives. Looking at the rate expression equation for *R1* written in the last column of this table shows that the rate of *R1* depends only on the current value of *x1*; meaning that the higher the current population of *x1* is, the more it influences *dx1* in a positive direction. The column for *R2* has two non-zero entries, a negative entry for

dx1 and a positive value for *dx2*. *R2* is a predation process and it causes the derivative of *dx1* to decrease while simultaneously causing the derivative of *dx2* to increase. It captures the interaction where *x2* consumes *x1* which decreases *x1* while simultaneously increasing *x2*. The processes directly influence the derivatives and indirectly cause the changes to observed values. The rate of this predation process depends on the product of *x1* and *x2*, meaning that when both populations are high, the predation rate will be high and if both are low then the rate of predation will be low. Each of the other predation processes in this model operates in the same way with their respective populations. Finally, *R7* represents the death of the top level predator in this eco-system and the rate of death depends only on the current population of the top level predator *x6*.

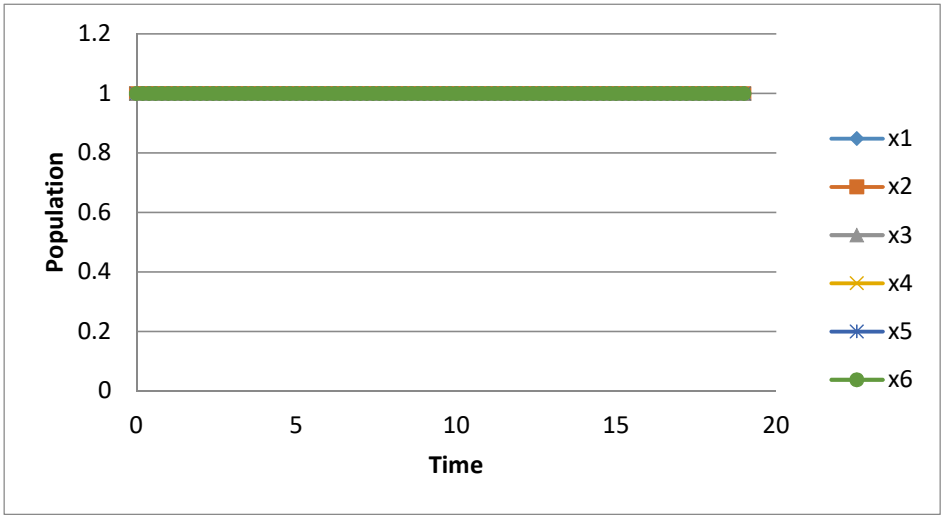


Figure 31 Trajectories of the model defined by Table 5 where initial values all start at 1.0

One of the aspects of this model that makes it easier to understand is that each derivative is influenced by only two processes. One process has a positive influence on the derivative while the other is negative. The balance of these two competing influences on each derivative accounts for the complex dynamics in a system of this nature.

The coefficient matrix and the process rates can be translated into the model equations by performing matrix multiplication between the coefficient matrix and the process rates as expressed in Figure 25. The equations along with the provided initial values are input into the RK4 simulation algorithm described in the previous section. The resulting trajectories of this set of model components are shown in Figure 31. Perhaps surprisingly, this model has no dynamics. All variables remain at their initial values of 1.0. This unexpected result can be easily verified by calculating the derivatives using the model equations in Table 5. The initial values are 1.0, so the rate equations all compute to 1.0. Thus the derivatives end up being the equation 1 – 1 = 0. All of the derivative values are zero and the system never changes. However, even a minor perturbation to any of the parameters anywhere will result in dynamic behaviour. For example, if the initial value of *x3* is changed from 1.0 to 0.5, it results in the trajectories shown in Figure 32. The trajectories are the result of the exact same model structure and parameters but using slightly different initial values.

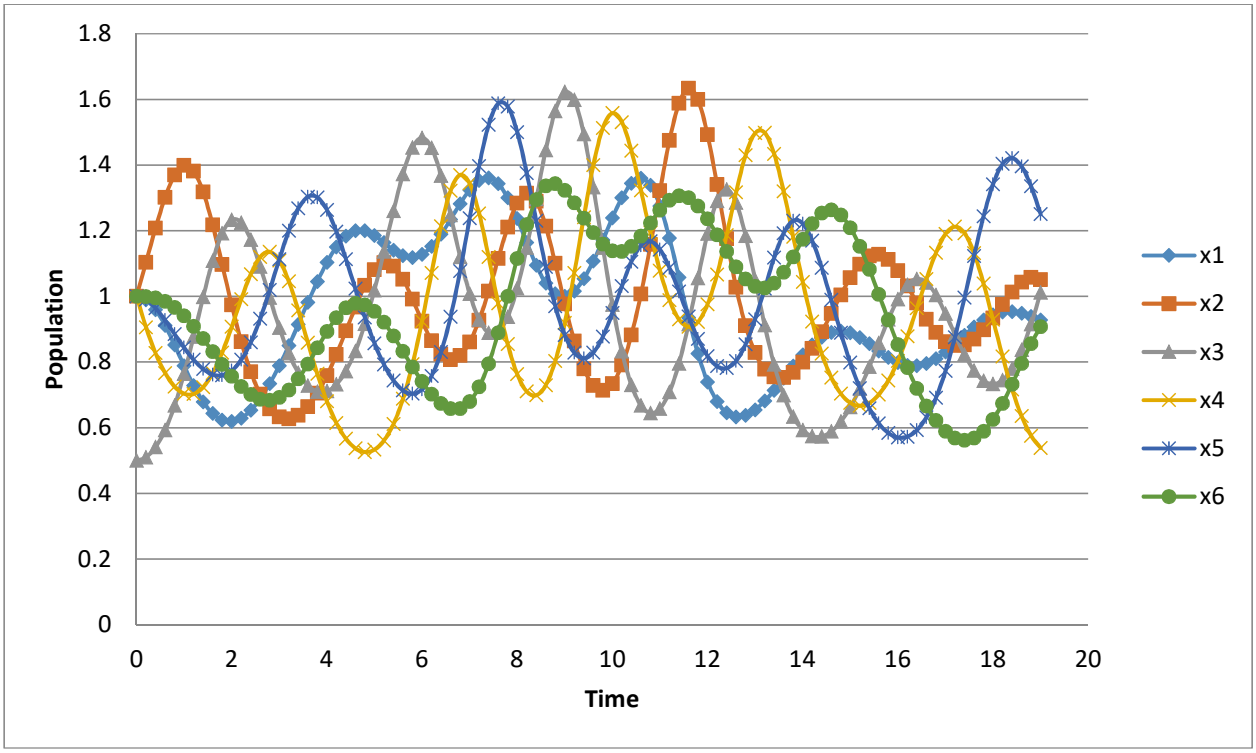


Figure 32 The model structure is identical to the one that produced the trajectories in Figure 31 but the initial value of the variable x3 was changed to 0.5, as can be seen on the grey line with triangle markers starts at 0.5 where all of the others start at 1.0.

One of the desirable outcomes for the trajectories is that they be uncoupled oscillations that are relatively steady but complex enough that their relationship is not immediately obvious. If the relationships were obvious that would negate any need for an automated discovery system. The trajectories in Figure 32 satisfy this criteria and on top of that, the process model is based on a known domain which can be encoded into a generic process library and incorporated into a discovery system. The chosen parameter values for the coefficient matrix in Table 5 are problematic. They were chosen as initial guesses because there are no simple guidelines for what qualifies as a good parameter, so 1 is as good a value as any other. The homogeneity of the parameter values makes this system appear to be implausible. Despite that, the values provide an order of magnitude starting point for tuning the parameter values to appear more realistic.

Table 6 A different coefficient matrix for the model structure defined in Table 5 that produces the trajectories in Figure 33

vars	R1	R2	R3	R4	R5	R6	R7
dx1	1.7	-0.8					
dx2		0.75	-0.95				
dx3			0.8	-0.9			
dx4				1.1	-0.8		
dx5					0.8	-1	
dx6						0.9	-1.1

Tuning the parameters resulted in the values in Table 6 and produced the trajectories in Figure 33 using initial values that were all 1.0. Initial values of 1.0 are reasonable in this case because they can be thought of as a fraction of some baseline population value, making the values plausible. The model structure, defined by the processes that appear in the model, are identical for the trajectories produced in each figure. Differences in parameter values or even initial values can change the trajectories. Changes in model structure will also result in different trajectories and various model structures from different domains will appear in future chapters.

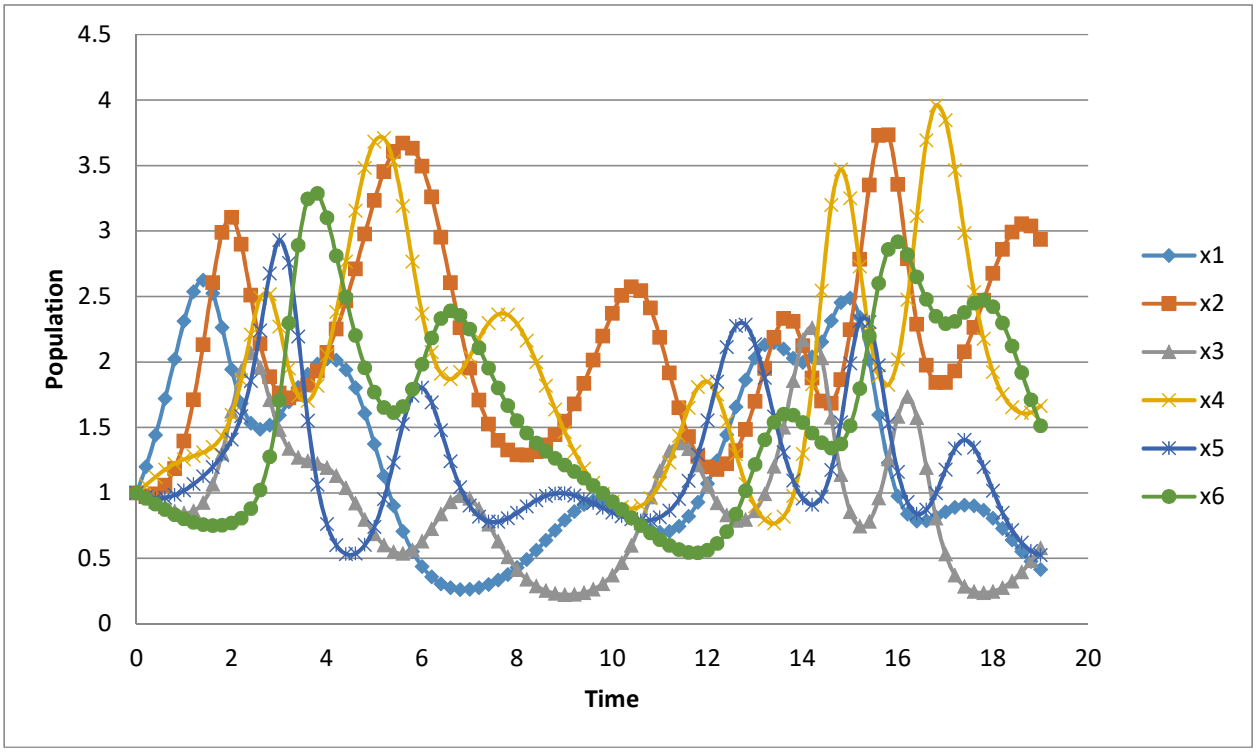


Figure 33 trajectories produced using the coefficient matrix from Table 6

5.5. Concluding remarks

Any discovery system will ultimately be faced primarily with empirical data but synthetic data sets play an important role during development. Synthetic data provides better control and allows us to use known models that serve as targets to help debug and test the performance of algorithms used during discovery. Creating these synthetic data can be a challenging task with open ended constraints and requires the user to perform many of the same difficult and tedious techniques that discovery systems are intended to automate. The user provides a set of processes along with all of the parameters and the initial values. The interaction between these components should produce both a plausible model structure and trajectories. Trajectories can be computed in the spreadsheet using the RK4 numerical approximation method, which provides a balance between accuracy and algorithm complexity. Expanding an existing model into a more complex by adding new, but similar, processes to the model helps to simplify the model building task. The next chapter describes an extension to the rate-based process model framework that builds on this idea of adapting an existing model for a new task.

6. Heuristic adaptation of rate based process models

A majority of the text from the following chapter originally appeared as part of the Third Annual Conference on Advances in Cognitive Systems on Atlanta, GA in 2015 (Arvay & Langley, 2015) . It describes an extension to the rate based process model framework that highlights an important aspect of the approach: the extension leverages the independence of each equation in order to facilitate model adaptation. Instead of building a model from the ground up, an existing model is used as a starting point and adapted to new data to explain a new situation. This suggests a reduction in the amount of intellectual effort required by the modeller as existing domain knowledge can be reused. We demonstrate computational benefits over constructing a model from scratch.

6.1. Introduction

In this chapter, we focus on the problem of *inductive process modelling* (Langley et al., 2002) where one is provided with multivariate time series data for some dynamic system and background knowledge about the types of processes that can occur in the domain. The goal is to generate a quantitative process model, including numerical parameters, that reproduces the observed trajectories and that predicts new values accurately. Such a model compiles into a set of linked differential equations, but it also accounts for the data in terms of unobserved processes. This distinguishes research in the area from work on differential equation discovery such as that by Džeroski and Todorovski (2008), which describes but does not explain them in deeper terms.

Earlier approaches to inductive process modelling, such as the SC-IPM system described in chapter three, generated many alternative model structures and then fitted their parameters to observations in order to evaluate the model. Parameter estimation involved repeated simulation of each model structure with different values and calculation of an error to drive gradient descent search, with random restarts to reduce the chances of finding local optima. This scheme was computationally expensive, scaled poorly to complex models, and even so did not always find good parameterizations.

These shortcomings of existing process induction systems lead to the development of a rate-based process model framework that greatly improves the computational efficiency while maintaining the explanatory power of processes. The initial implementation of this framework was demonstrated using a regression-guided process modeller (RPM), detailed in the chapter four. Like its predecessors, RPM organizes differential equation models into distinct *processes*. These identify aspects of the equations that must stand or fall together. One of the important insights to arise from the RPM system was the applicability of the rate-based process framework for model adaptation.

In many cases, scientists are less concerned with creating a model from the ground up than with adapting an existing model to a new setting. This may occur when they believe the system under study has changed, so that the model no longer fits observations as well as those for which they devised it. Or they may have developed the model to explain data from one area and find that it does not fare as well on data for an adjacent area. In either case, they may need to revise the model's parameters to address quantitative changes, or even need to alter the model's structure by removing, adding, or replacing some of its component processes. In the sections that follow, we describe one approach to the task of adapting a process model to explain data in a new setting. We start by introducing a new system for model adaptation that builds on the ideas that underlie RPM's successes. Next we report empirical studies designed to show that this new system operates as intended and that it offers efficiency gains

over inducing a model from scratch. Finally, we discuss previous work on model revision and discuss important future work.

6.2. An approach to process model adaptation

Adapting an existing process model to explain and describe new time series should be less computationally intensive than constructing a model from scratch. Rate-based process models of the type just described are particularly well suited to revision because they consist of equations that one can evaluate independently. This separation of model components makes it straightforward to determine which equations to alter and which ones to retain without changes. To explore this prospect we have developed APM, for *Adaptive Process Modeler*, a system that revises an existing process model to explain newly observed data when needed. The program is implemented in LISP and operates in three stages, which we now describe in turn.

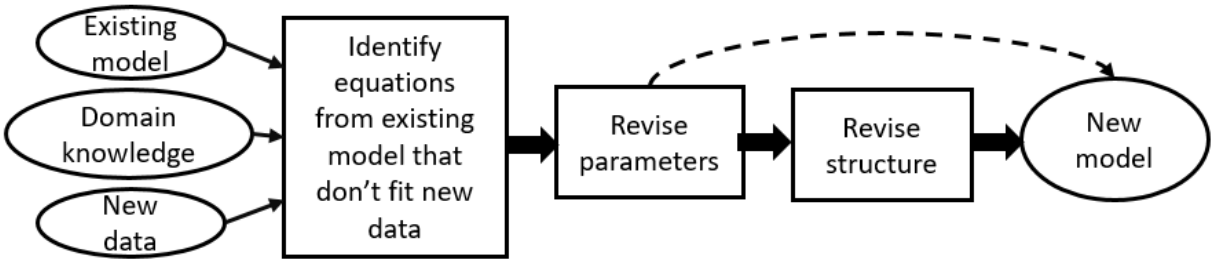


Figure 34 Workflow of the adaptation method. An existing model along with new data is provided and evaluated to identify anomalous equations. Parameters are revised to fit the new data and failing that, structure is revised.

6.2.1. Detecting Anomalous Derivatives

The first step in APM's model adaptation is to determine whether any equations require revision and, if so, which ones. This stage takes as inputs an existing base model, new time-series data to test it against, and criteria about acceptable deviations. The user must specify the correspondence between variables in the base model and new data set. The base model comprises a set of rate-based processes that maps onto a set of linear differential equations. This model predicts the derivative for each dependent variable on each time step, and these predictions are available for comparison to the corresponding 'observed' derivatives for each variable.

Table 7 (a) Quantitative process model for a six-variable predator-prey ecosystem and (b) the set of six differential equations into which the model compiles.

(a)

```
exponential_change[x1]
    rate        r = x1
    equations   d[x1] = 1.7 × r

holling_predation[x1, x2]
    rate        r = x1 × x2
    equations   d[x1] = -0.8 × r
                d[x2] = 1.3 × r

holling_predation[x2, x3]
    rate        r = x2 × x3
    equations   d[x2] = -1.4 × r
                d[x3] = 0.8 × r

holling_predation[x3, x4]
    rate        r = x3 × x4
    equations   d[x3] = -0.9 × r
                d[x4] = 1.1 × r

holling_predation[x4, x5]
    rate        r = x4 × x5
    equations   d[x4] = -0.8 × r
                d[x5] = 0.8 × r

holling_predation[x5, x6]
    rate        r = x5 × x6
equations[x5] = -1.0 × r
                d[x6] = 0.9 × r

exponential_change[x6]
    rate        r = x6
    equations   d[x6] = -1.1 × r
```

(b)

$$d[x_1] = 1.7 \times x_1 - 0.8 \times x_1 \times x_2 \quad (1)$$
$$d[x_2] = 1.3 \times x_1 \times x_2 - 1.4 \times x_2 \times x_3 \quad (2)$$
$$d[x_3] = 0.8 \times x_2 \times x_3 - 0.8 \times x_3 \times x_4 \quad (3)$$
$$d[x_4] = 1.1 \times x_3 \times x_4 - 0.8 \times x_4 \times x_5 \quad (4)$$
$$d[x_5] = 0.8 \times x_4 \times x_5 - 0.8 \times x_5 \times x_6 \quad (5)$$
$$d[x_6] = 0.9 \times x_5 \times x_6 - 1.1 \times x_6 \quad (6)$$

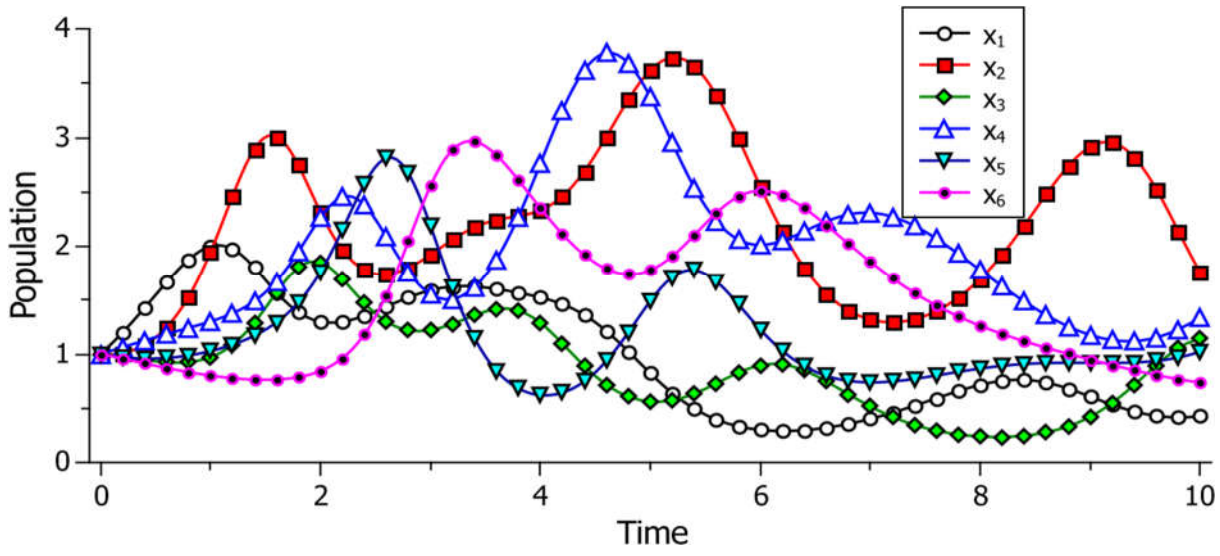


Figure 35 Trajectories for the six-variable predator-prey system from Table 7.

APM uses the r^2 statistic, a measure of the variance explained, as its criterion for detecting anomalous behaviour. We assume the system has access to the base model's r^2 score for each variable on the initial data set. The user specifies a fraction (e.g., 0.8 or 0.9) of the prior score that he would find acceptable for the model's fit to the new data set. The system tests the base model on the new data, calculates an r^2 score for each equation, and computes the ratio between the new score and the prior one. If this ratio falls below the user-specified threshold, APM marks the equation as a target for revision. Based on this analysis, the anomaly detection module outputs two sets of equations, one containing elements that need revision and another whose equations need not be altered.

As an example, suppose we have the base process model in Table 7(a), which compiles into the six differential equation in Table 7 (b) and which produces the trajectories in Figure 35. Further suppose that the user decides to reuse this model to explain the new trajectories in Figure 36, which are consistent with the process model and equations in Table 8. Note that three equations are identical in the two models, but the equation for $d[x2]$ differs in its parameters, while those for $d[x1]$ and $d[x3]$ differ in their structure. In this example, we can see that the equations for $d[x4]$, $d[x5]$ and $d[x6]$ are identical in both models. However, the trajectories for each equation in both models are very different. It may seem counter intuitive for identical equations with the same initial conditions to produce different trajectories, but remember that they are part of a fully connected system, so changes to any variable can influence all of the others. APM does not have any details about the model that generated the new data; it knows only the initial data set, the base model, and new data that it has been asked to explain.

Figure 37 plots the derivatives estimated from observed values against those predicted by the base model equations on the new data set. APM uses the observed values of variables to calculate process rates, which the equations then combine to generate predicted derivatives on each time step. A model that makes accurate predictions will have points that fall along the diagonal line, as with the equation for $d[x4]$. The agreement between observed and predicted derivatives is especially poor for $d[x1]$'s equation, reflecting a low r^2 score of -4.8. Suppose that, in this case, the user has decided that a revision threshold of 0.8 is appropriate and the ratio of r^2 scores on the old and new data sets exceed this threshold for equations 4, 5, and 6. APM retains those equations for the final model with no changes. However, suppose further that equations 1, 2 and 3 have ratios that fall below 0.8. This leads the system to mark them for revision during later stages of processing.

6.2.2. Revising Equation Parameters

Once APM has identified which equations, if any, require revision, it attempts to reestimate their parameters. Recall that each equation is stated as a linear combination of process rates, with the latter being parameter-free algebraic combinations of observed variables. Inputs to this module include a set of equations to be revised, the set of derivatives estimated from observed trajectories, and the generic forms of processes that appear in the base model.

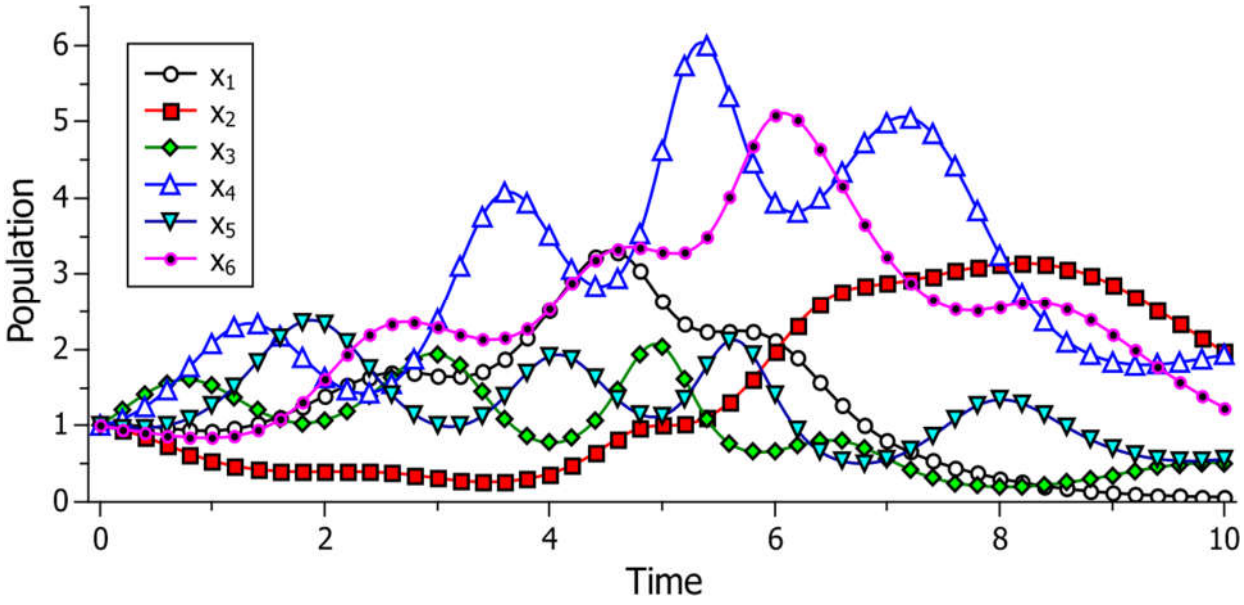


Figure 36 Trajectories for the six-variable target predator-prey model in Table 8.

The generic processes are necessary because they contain information about constraints on parameter values, such as whether they must be positive or negative. These constraints reduce the chances that regression analysis will find parameters that fit the data but do not generalize. The outputs from the parameter revision module are two sets of equations. One contains expressions with new parameter values that exceed the threshold for r^2 ratio between the initial and new data. A second set, possibly empty, contains derivatives for which APM could not find acceptable parameters using the original equation structure.

Recall that, in our example, APM has marked equations 1, 2, and 3, along with their associated derivatives, as requiring adaptation because their r^2 ratios were below the 0.8 threshold. In response, the parameter revision module invokes multivariate regression on each derivative to determine new parameter values, respecting the constraints in the generic processes. After this step, equations 4, 5, 6, and 2 are in the final model, while equations 1 and 3 remain in the revision set for future processing. In this case, APM cannot find acceptable equations for $d[x1]$ and $d[x3]$ because, in the true model (which it is attempting to induce), different structures determine these derivatives. As a result, parameter revision is insufficient to account for the new data so the system continues to the next stage.

Table 8 (a) Quantitative process model for a six-variable ecosystem that is the target for revision and (b) the set of differential equations into which the model translates. Differences from the base model in Table 2 are indicated in boldface font and underlined.

(a)

exponential_change[x1]
- rate: r = x1
- equations: d[x1] = 1.7 × r

holling_predation[x1, x2]
- rate: r = x1 × x2
- equations: d[x1] = -0.8 × r; d[x2] = **0.25** × r

holling_predation[x1, x3]
- **rate: r = x1 × x3**
- **equations: d[x1] = -0.8 × r; d[x3] = 1.1 × r**

holling_predation[x2, x3]
- rate: r = x2 × x3
- equations: d[x2] = **-0.7** × r; d[x3] = 0.8 × r

holling_predation[x3, x4]
- rate: r = x3 × x4
- equations: d[x3] = -0.9 × r; d[x4] = 1.1 × r

holling_predation[x4, x5]
- rate: r = x4 × x5
- equations: d[x4] = -0.8 × r; d[x5] = 0.8 × r

holling_predation[x5, x6]
- rate: r = x5 × x6
- equations: d[x5] = -1.0 × r; d[x6] = 0.9 × r

exponential_change[x6]
- rate: r = x6
- equations: d[x6] = -1.1 × r

(b)

$$d[x_1] = 1.7 \times x_1 - 0.8 \times x_1 \times x_2 - \underline{\mathbf{0.8} \times \boldsymbol{x_1} \times \boldsymbol{x_3}} \quad (1)$$
$$d[x_2] = \underline{\mathbf{0.25}} \times x_1 \times x_2 - \underline{\mathbf{0.7}} \times x_2 \times x_3 \quad (2)$$
$$d[x_3] = 0.8 \times x_2 \times x_3 - 0.8 \times x_3 \times x_4 + \underline{\mathbf{1.1} \times \boldsymbol{x_1} \times \boldsymbol{x_3}} \quad (3)$$
$$d[x_4] = 1.1 \times x_3 \times x_4 - 0.8 \times x_4 \times x_5 \quad (4)$$
$$d[x_5] = 0.8 \times x_4 \times x_5 - 0.8 \times x_5 \times x_6 \quad (5)$$
$$d[x_6] = 0.9 \times x_5 \times x_6 - 1.1 \times x_6 \quad (6)$$

6.2.3. Adapting Equation Structure

If APM decides that anomalous variables cannot be handled by parameter revision, it resorts to structural adaptation, which adds or removes processes from a model. This stage takes as inputs the results of parameter revision and the known generic processes. The latter may include processes that did not appear in the base model but that are plausible candidates for the domain. This module constructs a set of process instances that include the anomalous derivative terms. It also examines the processes that appear in the equations for other derivatives to determine if any of them must appear in the equations it is about to construct. Process instances often influence several equations can only be added or removed as a complete unit. Making changes to a nonanomalous equation by adding or removing a processes that influences both a nonanomalous equation and an anomalous one would alter the r^2 value that was previously declared to be acceptable, thus causing a new anomalous equation to that requires adaptation.

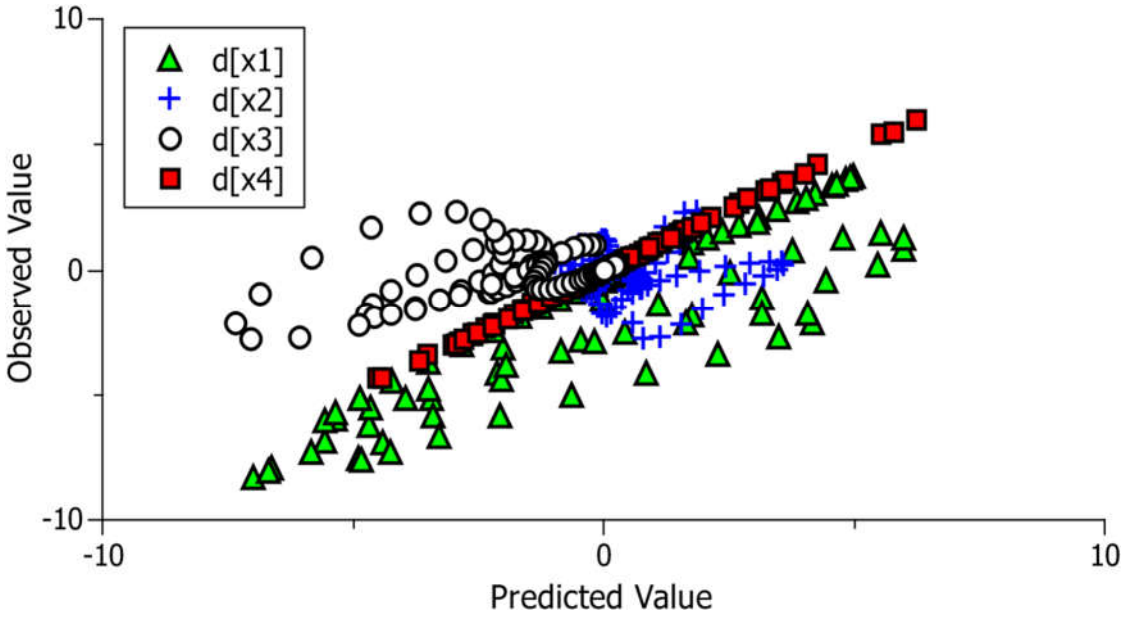


Figure 37 Observed vs. predicted derivatives for four variables from the model in Table 7. Acceptable equations have points that fall along the diagonal line, whereas ones that require revision are widely scattered. The reported r^2 scores for dx1 dx2 and dx3 were −3.13, 0.22, and −4.84.

The structure revision algorithm begins by removing all unnecessary processes from the model. Recall that a complete model is composed of several processes, each of which usually appears in multiple equations. Unnecessary processes appear only in equations that need revision. Removing these unnecessary processes will usually result in the equations that require revision containing some of the remaining required processes. These must appear because they are present in other equations that do not require revision so they cannot be completely removed from the model.

APM begins structure revision with the variable that has the most processes already present in the equation. This keeps the new equation consistent with the existing model and reduces the size of the search space because some of the processes are already present which reduces the number of possible combinations. Like RPM before it, the system carries out breadth-first search, ordered by complexity, for new equation structures. The starting structure for search contains only the process rates required due to their appearance in other equations. The module creates more complex equations by adding rate terms one at a time until it reaches a set maximum number of processes. Search terminates when an equation exceeds the threshold for r^2 ratios or it reaches a maximum number of processes. In cases where multiple equations at a given complexity level exceed the threshold, APM returns the best-scoring candidate. The final output is the set of equations found during search plus those produced by the previous stage.

Let us return to our example, with structural revision continuing where the parameter estimation procedure left off. Recall from the previous sub-section that APM had altered the parameters for the equation for $d[x2]$ from Table 8(b), but that this did not suffice for $d[x1]$ or $d[x3]$. Because the system must only find equations for these derivatives, it generates only those process instances that contain them as dependent variables. In response, APM carries out a structural search for equations that explain

their estimated values as a linear combination of process rates. When new process structures are required, APM will induce process instances using the same methods used by RPM, described in section 4.3, only inducing the process instances that are relevant to the variables that require revision. In addition, rather than starting from scratch, as in RPM, this begins with a partial model that includes the equations from earlier stages.

The process diagram in Figure 38 shows the initial model that is to be adapted to new data. The new process that will be required to account for the new data appears with a dashed outline in the figure. There are no new arrows connecting to $d[x2]$, $d[x4]$, $d[x5]$, or $d[x6]$, meaning that the structure for those equations is unchanged. The new predation process is connected to $x1$ and $x3$, which means it appears in the equations for $d[x1]$ and $d[x3]$. Furthermore the direction of the arrows indicate that the process has a negative coefficient for $d[x1]$ and a positive one for $d[x3]$. The specific values of the coefficients are found during parameter estimation.

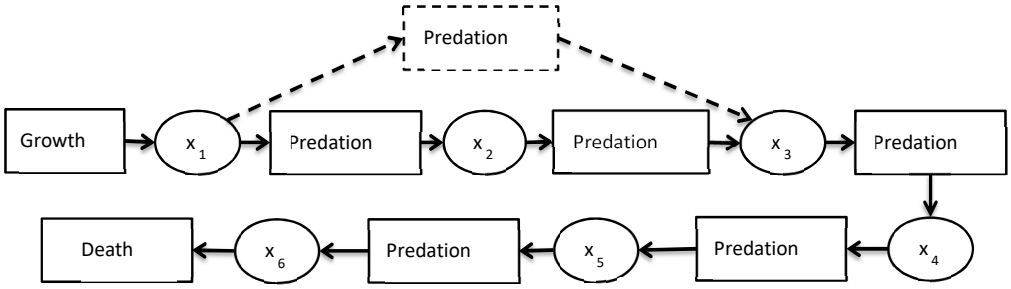


Figure 38 Process diagram for the six-variable target predator-prey model in Table 8.

The rate-based process framework assumes that any variable appearing in a rate expression must also be influenced by that rate. Thus, if the rate expression $x2 \times x3$ appears, then it must appear in equations for both $d[x2]$ and $d[x3]$. Examination of equations 2, 4, 5, and 6 from the base model in Table 7 reveal that the second equation contains the rate expression $x1 \times x2$, which means that same rate expression must appear in the revised version of equation 1. In addition, the appearance of the term $x2 \times x3$ in the second equation and the appearance of $x3 \times x4$ in the fourth equation (from Table 8) determine that both $x2 \times x3$ and $x3 \times x4$ must appear in the third equation. APM begins from the third equation rather than the first because it has two required rate terms, rather than only one. At the end of its structure search, the system returns the model in Table 8. This carries over equations 4, 5, and 6 from the base model, but the second equation has altered parameters, whereas the first and third have entirely new structures.

In summary, model adaptation in APM operates in three stages: anomaly detection, parameter revision, and structure adaptation. Each derivative is tested independently, which simplifies anomaly detection and lets the system identify which parts of the base model it need not change. Parameter revision involves the reestimation of rate coefficients using multiple linear regression with known terms. Structure revision requires more effort, but it takes existing equations and their constituent processes into account to determine the order in which to incorporate derivatives and to reduce the number of candidate structures considered. This approach to model adaptation builds on the strengths of the rate-based process framework discussed by Langley and Arvay (2015) to adapt existing models to new data in an efficient manner.

6.3. Experimental Evaluation

We have designed APM to take an existing base model and adapt it to explain new data. The system attempts to detect when equations should change, revise their parameters if possible, and to alter their structure if necessary. In this section, we report results obtained on synthetic data that demonstrate APM's basic abilities for parameter and structure revision across four distinct types of revision scenarios.

We report CPU times for each scenario compared against inducing a process model from scratch. We also demonstrate generality by evaluating APM on a more complex aquatic ecosystem domain. In addition, we present the results of a scaling study that compares APM's efficiency as the number of variables in the model increases.

6.3.1. Basic Functionality

Our initial evaluation aimed to demonstrate that APM has the intended ability to identify anomalies and revise the base model in response. We used six-variable predator-prey models similar to those in Table 7 to generate synthetic trajectories for testing purposes. Langley and Arvay (2015) report that RPM is robust to reasonable amounts (ten percent) of noise within its tested domain, so we did not add random variation to these data as they come from the same domain. We tuned parameter values to produce stable trajectories over the observed time frame. We provided APM both with the handcrafted base model and with appropriate generic processes that specified the algebraic forms for rate terms and constraints on parameters. We set the anomaly detection threshold to 0.8, which means that, to be acceptable, the model's r^2 on the new data must be no less than 0.8 of that on the original data.

In the first study we evaluated APM's ability to revise model parameters. We altered coefficients in the first two equations in the base model and used the result to generate new trajectories. In this case, the system calculated r^2 values of 0.56, -2.27, 0.99, 0.99, 0.99, and 0.99 on the new data. The corresponding r^2 scores on the initial data were all 0.99, which gave ratios of 0.57, -2.29, 1.0, 1.0, 1.0, and 1.0. The first two fell below the user-specified 0.8 threshold, so APM reestimated the parameters for these equations, which produced r^2 scores of 0.99 in each case. In response, the system incorporated them into the model it returned without resorting to structure revision. The r^2 score is calculated using the equation $1 - \frac{SS_{res}}{SS_{tot}}$. SS_{res} is the sum of squares of the residuals, which are the difference between the model predictions and the observations. SS_{tot} is the sum of squares of the difference between observed data and the mean of the observed data. A negative r^2 value can be interpreted to mean that the model predictions are worse than a mean value model.

Our second study evaluated APM's ability to adapt model structure. We created the target model by modifying the base model structure, adding one process that affects variables $d[x1]$ and $d[x3]$, then used the result to generate new trajectories. APM calculated r^2 values of -3.3 and -5.1 for $d[x1]$ and $d[x3]$, respectively, with all other values above the revision threshold. In response APM first attempted to revise the parameters for these equations, resulting in new r^2 values of 0.71 and 0.21 for $d[x1]$ and $d[x3]$. The resulting ratios fell below the revision threshold, so the system invoked its structure revision module. This produced new equations for $d[x1]$ and $d[x3]$ that included the correct new process and corresponding r^2 scores of 0.99. This result showed that APM can adapt model structure when parameter revision fails.

Our third run aimed to demonstrate APM's ability to handle cases that require both parametric and structural changes. Here we provided the system with the trajectories in Figure 36, which we generated using the target model in Table 8. In this case, anomaly detection noted that the r^2 values drop from 0.99 for all derivatives to -3.13, 0.22, and -4.84 for $d[x1]$, $d[x2]$ and $d[x3]$, with the others remaining very high. APM behaved as described earlier, first finding that the r^2 ratios for these derivatives are below the 0.8 threshold and then revising parameters for their equations to obtain r^2 values of 0.58, 0.99, and 0.19 for $d[x1]$, $d[x2]$, and $d[x3]$, respectively. The ratios for $d[x1]$ and $d[x3]$ were still below the 0.8 threshold, so the system attempted to adapt the structure of their equations, after incorporating the updated equation for $d[x2]$ into the new model. The structural revision module returned new equations for $d[x1]$ and $d[x3]$ (shown in Table 8), each with 0.99 as their r^2 score. These results demonstrate that APM first revises parameters and then resorts to structural adaptation only for equations that require it.

Our fourth run evaluated APM's ability to handle data sets that contain new variables. This adaptation task differs from earlier variants in that one must create equations from scratch for the new terms. We constructed a target model by extending the linear food chain of the base model with two new variables, $d[x7]$ and $d[x8]$, including a new process that influences $d[x6]$ and $d[x7]$ connecting them to others. APM began anomaly detection on the variables present in the base model, returning an r^2 of 0.56 for $d[x6]$ and high scores for the other variables. Parameter revision on $d[x6]$ returned an updated score of 0.62, which was still below the revision threshold. APM went on to revise the structure for $d[x6]$, returning a new equation with an r^2 value of 0.99. In this model, $d[x6]$ and $d[x7]$ are influenced by the same process, so once the system found the equation for $d[x6]$, a portion of the equation for $d[x7]$ had also been determined. APM then induced an equation for $d[x7]$ before moving on to $d[x8]$, the final variable. The resulting model included two new equations, one revised equation, and five carried over from the base, all with r^2 scores of 0.99. This study demonstrated APMs ability to add equations for entirely new variables while adapting others as necessary. Each study demonstrated a specific revision ability but any combination of them can occur together, such as different parameter values along with new variables and generic processes.

6.3.2. Generality of the Approach

In order to demonstrate APM's generality, we repeated the previous experiments in a more complex setting based on an aquatic ecosystem model described by Bridewell et al. (2007). Figure 39 shows the process diagram for the base and target model. This model involves more interaction among variables than the predator-prey food chain models used in the previous section. Predator and prey species still appear in this ecosystem, but so do several other variables, and some processes have more than one input or output. This model's structure relates five variables and five processes through and extended feedback loop.

The aquatic ecosystem model has phytoplankton as a primary producer that increases via a growth process, absorbing the nutrients iron and nitrate. Zooplankton consumes phytoplankton through a predation process that reduces the phytoplankton population, increases the zooplankton population, and produces detritus. Zooplankton population decreases through a death process that also produces detritus. Remineralization reduces detritus and produces iron and nitrate, while an external source of nitrate also influences nitrate concentration. The feedback loop closes as the growth process consumes the iron and nitrate to produce phytoplankton. Parameter values were tuned so that the model tends toward steady-state values for all variables, as shown in Figure 40.

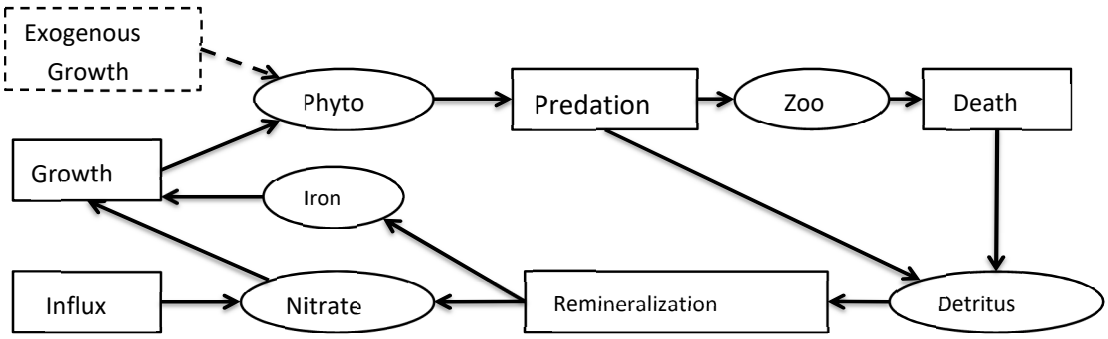


Figure 39 Process diagram of the aquatic ecosystem model. The target model includes an exogenous growth process, indicated by dashed outlining, that influences the phyto variable.

We changed the target model by adjusting the parameters in the nitrate equation and by adding the exogenous growth process that influences only the *phyto* variable. The rate equation for the exogenous growth process includes the exogenous variable. The purpose of including a new term is to demonstrate APM's ability to handle exogenous variables and to produce very different dynamics in the target data, as we wanted to show that the system identifies and retains appropriate parts of the model despite very different trajectories. As Figure 40 shows, the target data shows overall long term stability with short term oscillations about an equilibrium value and the modelling system is able to reproduce these dynamics in the adapted model without issue.

We evaluated APM's ability to adapt this more complex model structure starting from the base model in Figure 39 (less the exogenous growth process) and given the trajectories in Figure 40. Adaptation proceeded in the same manner as before. APM first checked the fit of each equation in the base model on the new data, determining that the equation for $d[phyto]$ and the equation for $d[nitrate]$ have r^2 values of 0.47 and -1.92 while those for other equations were high. These two values fell below the user specified ratio of 0.8, so the system estimated new parameters for those equations. APM found that the new r^2 values for $d[phyto]$ and $d[nitrate]$ were 0.65 and 0.99, the latter being high enough to be included in the new model. After this, it went on to search for a new structure for $d[phyto]$ in the same manner as the previous study, eventually returning an r^2 of 0.99.

Adaptation in this setting works in the same manner as in the simpler one. More complex models with processes that influence more variables are handled just as easily. The system detects anomalous derivatives and adapts them as necessary, first by altering parameters and then changing structure, if required, by adding new processes or new variables if appropriate. The approach treats each equation independently, so complexity in the overall model structure need not introduce complexity in individual equations, which APM still finds using linear regression. These runs provide evidence that APM can take an existing process model and adapt it as needed to explain new trajectories in a variety of different circumstances. They show that the system supports the basic abilities that we intended, but we are also interested in other aspects of its behaviour.

Table 9 (a) Quantitative process model for the base aquatic ecosystem, with differences between the base and target model are indicated in boldface text and underlined. (b) The set of differential equations into which the model translates. Equation 3T shows the parameter values used in the target model equation

(a)

```
Growth[phyto, iron, nitrate]
    rate          r = phyto × iron × nitrate
    equations     d[phyto] = 4.0 × r
                  d[nitrate] = -2.8 × r
                  d[iron] = - 1.15 × r
predation[phyto, zoo, detritus]
rate              r = phyto × zoo
    equations     d[phyto] = -1.9 × r
                  d[zoo] = 1.5 × r
                  d[detritus] = 1.0 × r
death[zoo, detritus]
     rate         r = zoo
     equations    d[zoo] = -0.75 × r
                  d[detritus] = 0.64 × r

remineralization[detritus, nitrate, iron] rate
                  r = detritus
     equations    d[detritus] = -1.28 × r
                  d[nitrate] = 0.20 × r
                  d[iron] = 0.33 × r
influx[nitrate]
     rate         r = 1/nitrate
     equations    d[nitrate] = 0.5 × r
exo_growth[phyto]
     rate         r = phyto × exo
     equations    d[phyto] = 1.2 × r
```

(b)(1) $d[phyto] = 4.0 \times phyto \times iron \times nitrate - 1.9 \times phyto \times zoo + \underline{\mathbf{1.2} \times \boldsymbol{phyto} \times \boldsymbol{exo}}$

(2) $d[zoo] = 1.5 \times phyto \times zoo - 0.75 \times zoo$

(3) $d[nitrate] = -2.8 \times phyto \times iron \times nitrate + 0.2 \times detritus + 0.5 \times \frac{1}{nitrate}$

(3T) $d[nitrate] = \underline{\mathbf{-3.2}} \times phyto \times iron \times nitrate + \underline{\mathbf{0.4}} \times detritus + \underline{\mathbf{0.3}} \times \frac{1}{nitrate}$

(4) $d[iron] = -1.15 \times phyto \times iron \times nitrate + 0.33 \times detritus$

(5) $d[detritus] = 0.64 \times zoo + 1.0 \times phyto \times zoo - 1.28 \times detritus$

6.3.3. Computational Benefits

We argued earlier that adapting an existing process model to new data should be more efficient computationally than constructing a new model from scratch. Our approach to parameter revision, which calls multivariate regression, requires no search at all, so these benefits should be largest when only changes to model coefficients are needed, but there should also be savings when structural changes arise. The reductions for parameter revision are due in part because it sidesteps calculation of unnecessary process rates. APM's use of linear regression to estimate parameters depends on these rates being calculated from observed values on each time step. To this end, the system calculates the process rates that appear in the equations for that model structure. When structural adaptation is necessary, there are still computational savings, as the size of the space is reduced substantially because only a subset of rate terms (ones that mention the derivative) are relevant. This decreases the number of process instances generated, thus decreasing the combinations that can appear in an equation. In addition, APM uses existing equations to constrain structural revision by keeping revised equations consistent with them.

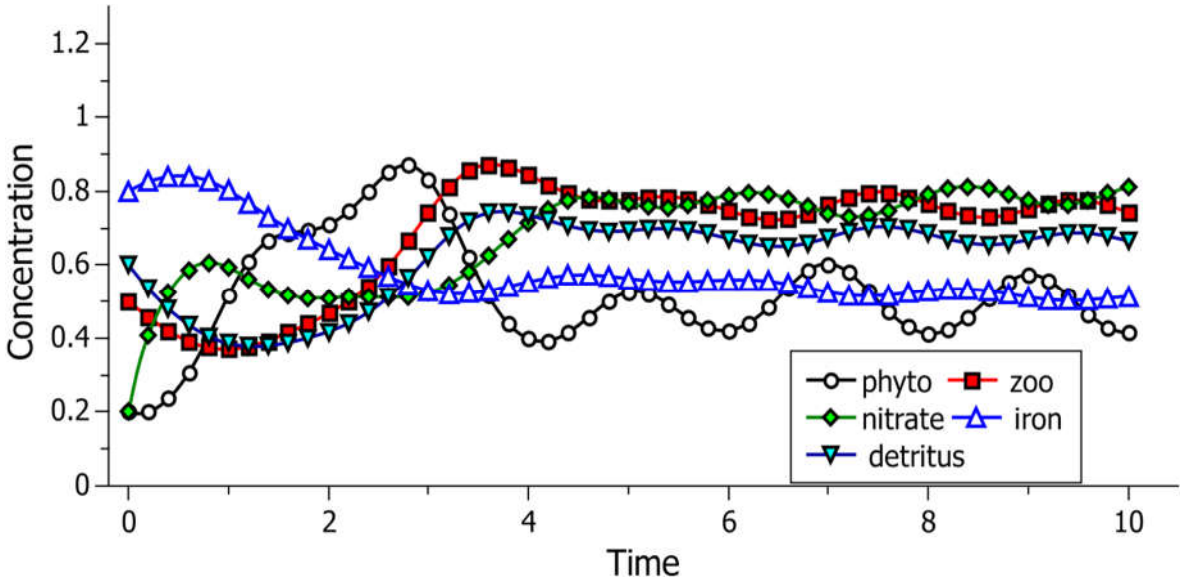


Figure 40 Trajectories of the target aquatic ecosystem model from Table 9

These computational savings are evident when comparing the CPU time needed for each revision run in section 6.3.1 with the time to construct a model from scratch using the older RPM system. These studies used the six-variable predator-prey model from Table 7 as the base model. In cases where only parameters need to be estimated, APM revised the base model to produce the target model in 1.2 ms, while RPM requires 19.1 ms (averaged across 50 runs) to produce the target model. When structural adaptation is necessary, APM adapts the base model in 5.5 ms while RPM requires 29.4 ms. Similar results emerge when both structure and parameter adaptation are required, with APM taking 5.5 ms and RPM taking 28.7 ms. In the final case, when two new variables are introduced, APM finds a revised model in 8.9 ms while RPM requires 32 ms. In all scenarios, revision is substantially faster than searching from scratch, with the greatest benefit occurring when only parameter revision is necessary.

Moreover, this benefit will increase as more of the original model is retained. This is shown by adding variables to the base model while keeping constant the number of variables that differ in the target model. Figure 41 shows the CPU time to construct a model from scratch using parametric revision and using structural revision. The first revision scenario from section 6.3.1 corresponds to parametric revision and the second involves structural revision. In each case, the structure, parameters, and fits of the final models produced by RPM and APM are identical despite the differences in processing time.

Most importantly, the amount of benefit increases as APM retains more of the base model. RPM takes an average of 74 ms to construct a ten-variable model from scratch. Revising an existing model that has eight retained equations is roughly 45 times faster for parameter revision and 12 times for structure. For a 20-variable model the benefit is even greater. Revising an existing model with 18 variables retained is about 87 times faster when only parameters are estimated and some 44 times when structure revision is necessary. Both revision schemes show linear growth with the number of variables, while RPM appears to be polynomial in this factor.

We also find computational benefits when the system revises the more complex aquatic ecosystem model. The process library for this domain generates 43 process instances. RPM requires 555 ms to construct this five variable model from scratch by checking roughly 1,400 equations. In contrast, APM revises the parameters in one equation and the structure of another equation in 4 ms. In this case, only one equation required structural revision, so only processes that influenced it were considered, greatly reducing the number of candidate elements included in the revised structure.

In summary, there are substantial computational savings when adapting an existing rate-based process model to explain new data, rather than inducing one from scratch. Parameter revision is the most efficient means, since it does not require search. When search for new equation structures is needed, the space is drastically reduced because it involves only a few variables and because the new equations must remain consistent with existing ones. As expected, the savings increase when more of the original model is retained, showing that revision of an existing model is far more scalable than constructing one from scratch.

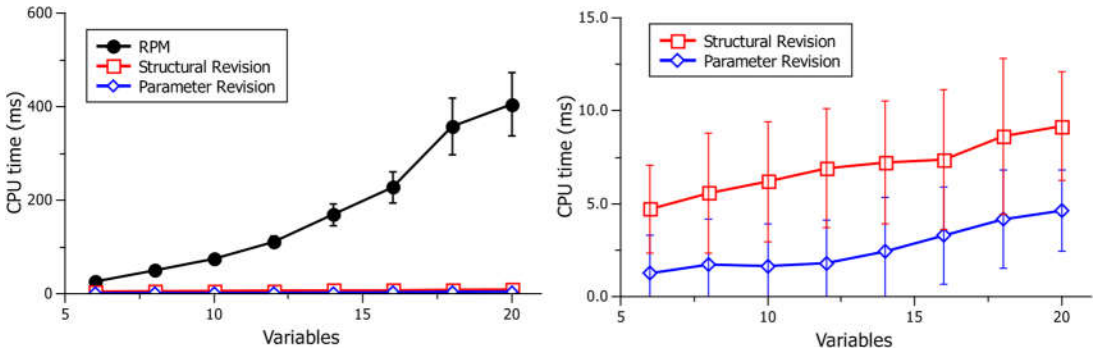


Figure 41 APM's processing time in milliseconds as a function of the number of variables in the target model when equations for variables require parameter revision and structure revision. The higher third curve in the left chart gives the time needed for RPM to induce the same models from scratch. The right chart shows a zoomed in view of the two revision curves.

6.4. Related Research on Model Revision

Our approach to adapting rate-based process models shares many ideas with previous efforts. For instance, the term *theory revision* has been used to describe a range of techniques that alter an existing model in response to new data, especially in the context of classifier learning. Mooney and Richards (1992) report one such method for automatically repairing a handcrafted logic program to cover supervised training cases. Towel, Shavlik, and Noordewier (1990) report another scheme that translates a logic program into a multilayer neural network, trains it on supervised data, and converts the result back into rules. These efforts focus on improving the classification accuracy of qualitative logical models, whereas APM instead addresses revision of explanatory scientific models for quantitative dynamic systems.

Another line of research, more closely related to our own, has applied similar ideas to revising explanatory scientific models. Darden (1990) proposes techniques for distinguishing different types of anomalies with respect to gene theory, which in turn have implications for revision. O'Rorke, Morris, and Schulenburg (1990) report a system that uses abduction to form hypotheses that it can revise if it encounters contradicting observations. Rajamoney (1990) describes an approach to explanation-based revision that changes an initial qualitative process model to assimilate anomalous observations. Each of these efforts involve the revision of explanatory models, but they focus on qualitative accounts rather than quantitative ones, as we have done.

A third paradigm for model adaptation draws on structural analogy. This technique uses existing knowledge about one situation to understand and make inferences about another. Falkenhainer (1990) reports a system that uses analogies with known process models to explain new scientific phenomena, although its adaptation abilities are limited. Friedman and Forbus (2010) expand on these ideas further, describing an explanation-based approach to conceptual change in process models that responds to new observations. These techniques revise qualitative models of scientific data, whereas our work has focused on adapting quantitative models.

Several researchers have developed systems that revise quantitative models using scientific data. Both Todorovski et al (2003) and Saito and Langley (2007) describe techniques for equation discovery that alter parameters and revise functional forms. In other work, Bay, Shrager, Pohorille, and Langley (2002) report a system that starts with a partial model of gene regulation stated as a linear causal model and then uses statistical regularities to alter its structure. These systems revise quantitative scientific models, but only for static scenarios, and they emphasize descriptive summaries of data rather than deeper explanations.

We use a simple threshold based on the r^2 values of the new and old data sets to detect anomalies. This is similar to drift detection methods described by Gama, Medas, Castillo and Rodriguez (2004) that also uses thresholds to analyse streaming data, including the amount of data to consider. It is also similar to concept drift (Gama, Žliobaite, Bifet, Pechenizkiy, & Bouchachia, 2014) where the relationship between input data and underlying model changes over time.

Bifet and Gavalda (2007) use adaptive windowing to automatically adjust the number of data points used to monitor a data stream in order to learn a new Bayesian model, improving both detection sensitivity and computational efficiency. These methods involve detecting anomalous classifications while our system is concerned with identifying anomalies in regression equations.

Some prior work on inductive process modelling has addressed the adaptation task. Asgharbeygi, Langley, Bay, and Arrigo (2006) describe a system that begins with a quantitative process model that it revises in response to observed time series. Like other early research in this area, their approach assumes a less constrained notion of process that does not partition rate expressions from derivatives, so it carries out a more extensive search through the space of model structures. Their system also relies on gradient descent search to estimate parameters. These make it both more computational expensive and less robust, due to the stochastic nature of the initial guesses, than the approach we have taken.

In summary, model adaptation has been studied in a number of guises. Some research has focused on logic programs for classification, while other efforts have dealt with qualitative process models. Most work on revising quantitative models has emphasized algebraic rather than differential equations. In contrast, the APM system uses the formalism of rate-based process models to partition the adaptation task across derivative terms, which considerably simplifies anomaly detection, parameter reestimation and structure alteration.

6.5. Concluding Remarks

In this chapter, we presented a novel approach to the adaptation of rate-based process models and their implementation in the APM system. We described this system in terms of three main activities: detection of anomalous derivatives, parameter reestimation, and structure modification. The first stage detects anomalies by comparing initial r^2 scores for each equation to r^2 scores calculated using new data. APM marks equations for revision if the ratio of r^2 scores falls below a threshold. Revision begins with parameter estimation, using multiple linear regression, followed by structural revision when this does not improve the fit sufficiently. Structural revision generates new process instances only for variables that require revision and starts with the equation that has the most processes already defined. These both reduce the size of the search space over that when inducing a model from scratch.

We demonstrated, using synthetic data from two domains, a six-variable linear food chain and a five-variable ecosystem with a feedback loop, that APM can successfully adapt a base model to explain new time series. Experiments also revealed the computational savings of adapting a model versus constructing one from scratch using RPM, an earlier system that also creates rate-based process models. These showed that the reduction in CPU time is greatest when only parameter estimation is needed, but there are even savings when structural adaptation must occur. Most of this benefit comes from a reduction in the search space, as attention is limited to process instances that include the anomalous derivatives. The times needed for both forms of revision appear to grow linearly with the size of the base model; when starting from scratch, the time instead grows as a higher-order polynomial.

One of the primary challenges for both RPM and APM involves the exhaustive search though the space of alternative model structures. The parameters for each equation are estimated without the need for simulation and in cases where rate equations are parameter free or contain known parameters, linear regression can be used to estimate coefficients. These two features contribute to greatly reduce the computational demand. However, the search for equations still grows as a higher order polynomial and can quickly become infeasible to calculate for complex domains with many process instance combinations. The following chapter tells the story of the development of a selection heuristic that identifies more promising process instances without resorting to an exhaustive search through all equation combinations.

7. Induction of rate based process models using a selection heuristic

The RPM system provides an approach computational scientific discovery that is efficient and robust to noise. It produces models that accurately account for, and explain, complex dynamics in terms of processes that connect to background knowledge. The rate-based process modelling framework independently evaluates each equation in a model. This allows the approach to naturally accommodate model adaptation which further improves the efficiency of finding a model. However, despite these advancements there are several important drawbacks to this approach, and this chapter aims to address one of the most significant—the exponential growth of the model structure search space.

Although the regression based parameter estimation algorithm is much more efficient than previous method, it still operates on a model space with exponential growth. No amount of efficient evaluation of individual equations can overcome an exponential growth in the number of equations that need to be evaluated. The complexity analysis of RPM presented in chapter four demonstrated that the growth in the search space is driven by the need to test all possible combinations of process instances. In most cases only a small subset of process instance combinations will fit the data well. The approach employed by RPM is to naively try all combinations, starting from the fewest terms and growing the number of terms.

There are several potential ways to address this problem. One is to reduce the size of the search space either by applying strict constraints on the processes that can appear together in a model or limiting the complexity of the domain. This would have the effect of reducing the number of process instances that can appear in the search space, thus reducing the number of possible combinations. This would have the largest impact on the size of the search space, but is also very difficult because constraints may not be obvious. In short, it is difficult to identify what constraints are appropriate to include. If inappropriate constraints are included it may prevent the system from finding a model that can account for the data. Another possible approach is to simply use faster computers. That may help this approach to scale to more complex systems but may also require access to state-of-the-art high performance computing (HPC) platforms and large number of core-hours. This technology usually has an associated financial cost that is can be a significant limitation for many research agendas. Another approach involves developing a selection heuristic that will replace exhaustive search. The selection heuristic should allow us to evaluate individual process instances or subsets of them from the space of possibilities and select the most promising ones for further evaluation. The difficulty in this approach is that it requires developing a technique that can characterize process instances as well as a ranking method.

Two different selection heuristics are explored in this chapter. The first is a forward selection heuristic based on Fourier analysis (Bloomfield, 2004). The process rates are observed and produce unique continuous trajectories. Each process instance trajectory can be decomposed using Fourier analysis into independent frequency-based components that can be used to characterize each process instance. The frequency-based characterization can then be used to select for more promising candidate processes to include in the regression algorithm. The second approach is a backwards selection heuristic that expands the application of the existing regression algorithm. The current search algorithm in RPM uses a forward search with no selection heuristic. It starts from a single process instance in each equation and, adding one at a time, tries all combinations until it finds a combination of process instances that fit the data well. In principle, backwards selection would begin by including all possible process instances

in an equation then the selection heuristic would be able to identify which process instances are irrelevant and drop them from the equation, leaving behind an equation that contains only relevant process instances.

7.1. A Fourier analysis based selection algorithm

Each process instance has an associated trajectory that can be computed from the observed data because the rate expression contains only observed variables and known parameters. The derivative is also computable from the observed data and it is assumed that the derivative is a linear combination of process instances. When looking at the process instance trajectories R1 and R2 in Figure 42, as well as the trajectory of the derivative of x1, they all have the appearance of a wave pulse. This suggests that a Fourier analysis could provide a means of characterizing the wave forms by transforming them into a set of independent components. Fourier analysis is a method for approximating any arbitrary wave form as a linear combination of sine waves of various frequencies. It decomposes any wave form into a set of simpler independent wave forms. The characterization of each process rate provides a systematic way to differentiate between process rates and has the potential to be used as the basis for selecting more promising process instances. The hypothesis is that because the target derivative is a linear combination of the relevant process instances, several frequency components will be shared between them, whereas the irrelevant process instances will share few if any frequencies with the target derivative. This technique is often used for periodic signals. In cases where the exact same signal is repeated, the same information should be able to be extracted from a single wave pulse.

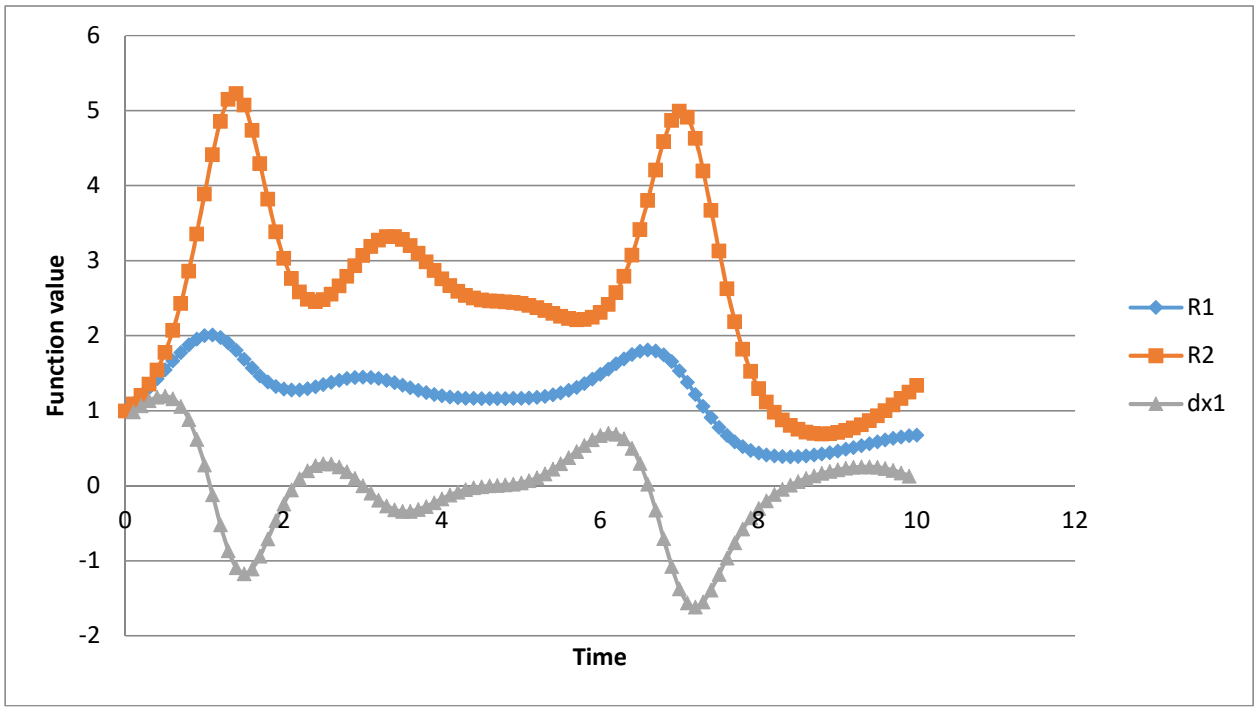


Figure 42 The wave-like appearance of the process instance trajectories R1 and R2 that combine linearly to generate the trajectory of the derivative dx1.

The data set used to test this hypothesis is the predator-prey system with 20 variables that first appeared during the evaluation of APM in chapter six. This synthetic dataset provided a set of known

relevant and irrelevant process rates that could be analysed to test the hypothesis that relevant process instance rates will share components while irrelevant ones will not. In order to perform the required analysis, the fast Fourier transform (FFT) function built into the software package R was utilized. The function takes as primary input a series of observations of a wave pulse and returns a corresponding set of complex numbers that represent the frequency, amplitude, and phase of each component of the observed wave pulse. The implementation uses the discrete Fourier transform which is often used for finite sample sizes with equal spacing. The results of the analysis were converted into amplitudes by taking the magnitude of the reported complex number and plotting a frequency histogram organized into harmonics that are equivalent to a specific frequency. This analysis is not concerned with absolute frequency values, only that ones that are shared between process instance and derivative transformations.

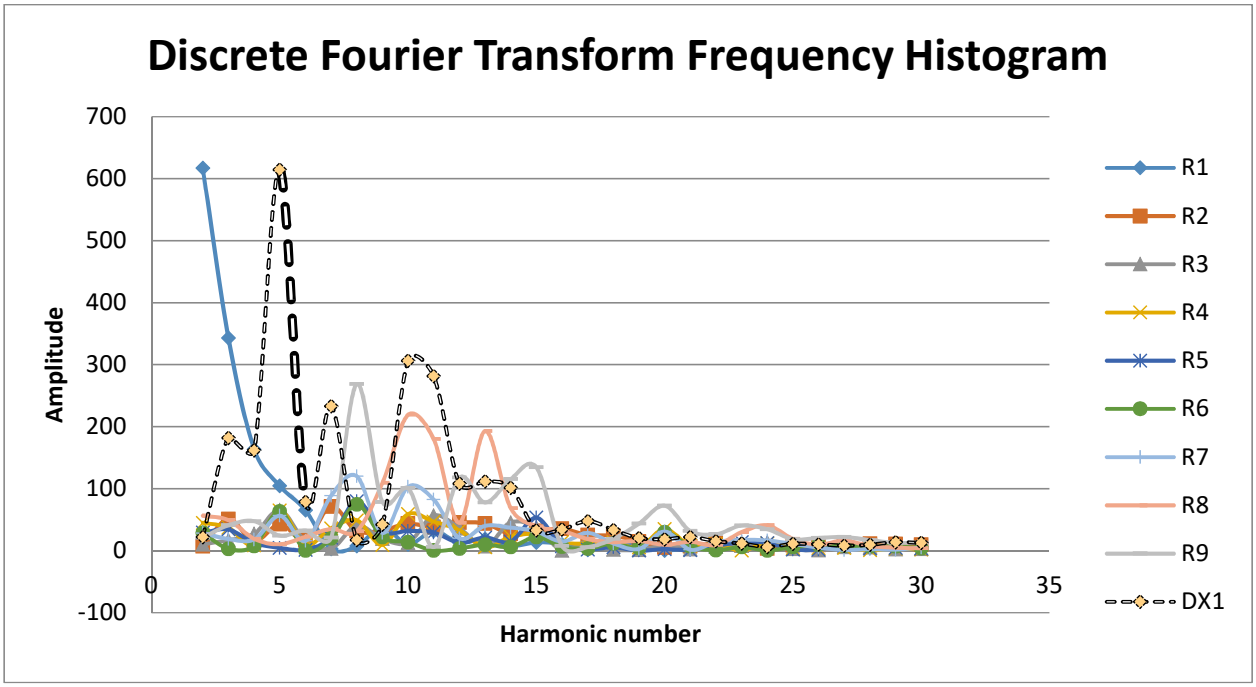


Figure 43 The Fourier transform frequency histogram of the first 9 process instances and the derivative of x1. Lines connect each histogram bin. The copious amounts of information provide a demonstration of the difficulty of extracting useful information from a Fourier transform of the data.

Figure 43 shows the results of a Fourier analysis of the trajectory of dx1 (from Figure 42) along with 9 process instances (R1 through R9) that appear in the model. A higher amplitude for a given harmonic means it is a more significant component of the signal. The harmonic number is another way of expressing the frequency of the component sine wave. The frequency can be calculated from the harmonic by dividing the length of the signal by two raised to the harmonic number. The first harmonic shown in the figure is harmonic number two which corresponds to a frequency that is $1/2^3$, or 1/8th, of the length of time of the signal provided for analysis. The information used to create this graph can be used to reconstruct the original signal of each wave pulse. Each harmonic represents a wave of a particular frequency and the amplitude is the equivalent to the contribution of that harmonic. If each of the components in the frequency distribution were added together it would reconstruct the original wave pulse.

Higher harmonic numbers correspond to higher frequency waves, which oscillate more quickly. The data shows that high frequency signals are not a major component of these signals because harmonics with high numbers are near zero amplitude. The majority of the higher frequency values are truncated from the graph in order to remove irrelevant information. Harmonics 0 and 1 are not plotted because the 0th harmonic represents an amplitude offset which accounts for the average value of a periodic signal and in this analysis that information is not relevant. Harmonic number one represents a frequency of a sine wave that has a period equal to the length of the original data sample window. Again this information is irrelevant because within this sample window there is one wave pulse, so this first harmonic will always have a high amplitude. Thus it was discarded because it does not differentiate anything in this data.

The hypothesis is that the target derivative will share high amplitude harmonics with the process instances rates that it is composed from in the model. If this hypothesis were true then the frequency distribution of DX1 should share peaks and valleys with distributions of R1 and R2 because DX1 is a linear combination of those two process instances. However, the results of the analysis shown in Figure 43 do not seem to support the hypothesis.

Figure 44 shows more detail of this analysis by removing all irrelevant process instances and scaling the vertical axis. It is clearer in this graphic that it would be difficult to design an evaluation algorithm based on this information that could determine if R1 and R2 are any better suited as candidate process instances than any other. The target has prominent peaks at harmonics 5, 7 and 10 and 11. R1, the blue line, appears to share only peak 11. R2 is more encouraging because it shares a peak at 5, 7 and 10. However, many of the other irrelevant processes also share a similar number of peaks in the harmonic amplitudes. The ideal scenario would be that when the target derivative has high amplitude the components would also have high amplitude and the relationship would also hold when the amplitudes are low.

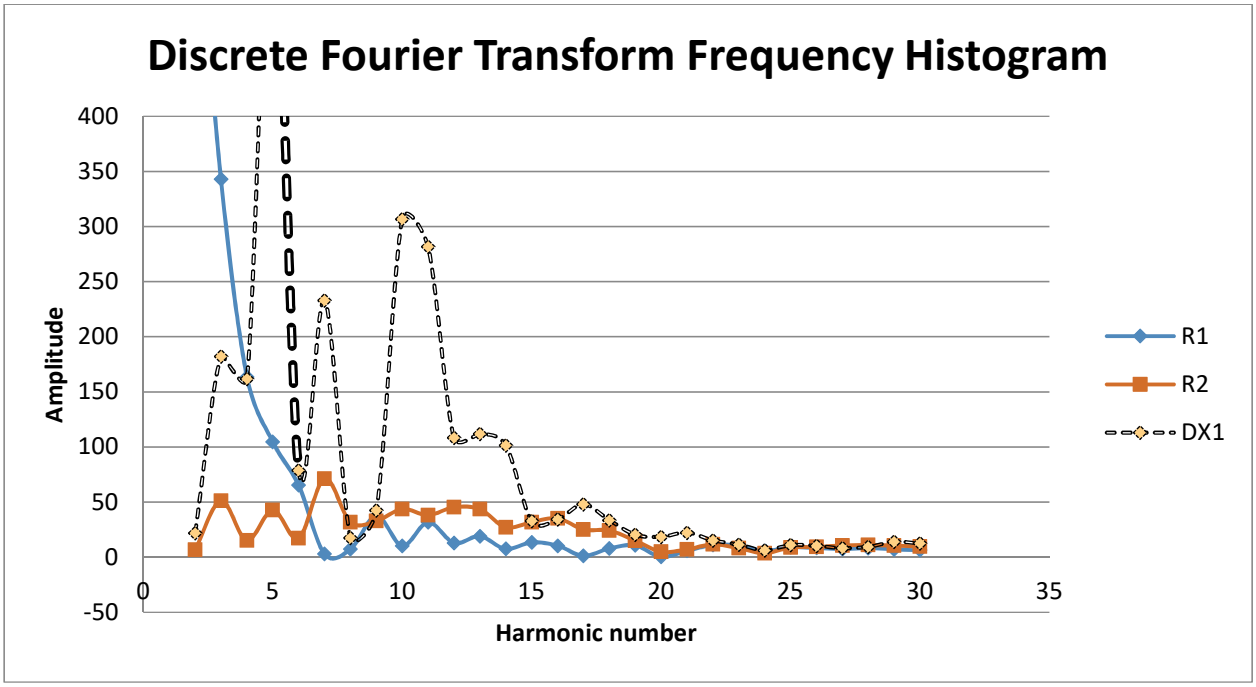


Figure 44 The Fourier transform of the five process instance trajectories along with the derivative of dx1 which shows more detail

Figure 45 shows a similar analysis as in Figure 44 but with the amplitudes normalized to 1.0 and the transformation of the derivative of x2 shown. The normalization enhances the smaller peaks to make it easier to compare components to the target. In this case the model structure states that R2 and R3 would be expected to share peaks with the target derivative. The data does not make it clear how to design an algorithm that could take advantage of this data to differentiate the relevant rate terms, R2 and R3, from the other irrelevant rate terms. Designing an algorithm that could reliably determine that R2 or R3 would be a better correspondence with DX2 seems unlikely. For example R4, the yellow line, seems to follow approximately the same pattern as DX2 for the first seven harmonic values, however we know from the model that R4 is irrelevant with respect to DX2. We also know that R2 is relevant, but its pattern shares very few, if any, peaks. Finally, R3 is also relevant and it shares some similarities.

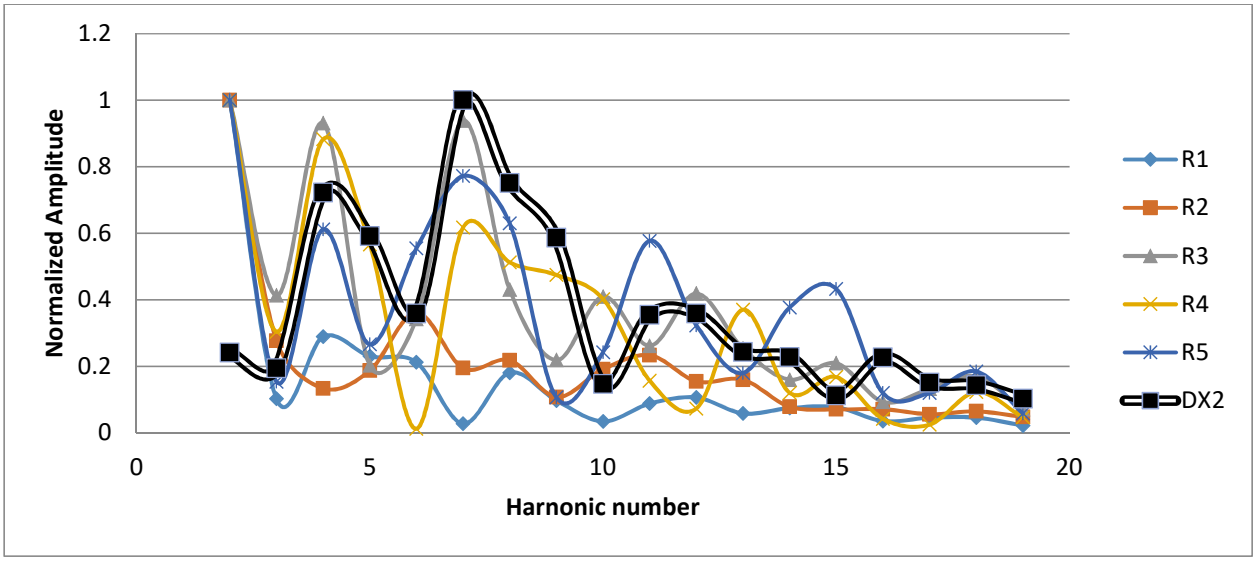


Figure 45 A normalized Fourier transform frequency histogram of several process instances along with dx2

There are several challenges that have become apparent in this approach. The most important is that decomposing the waveforms into frequency based components and looking at amplitudes does not seem to provide a useful way to differentiate between relevant and irrelevant process instances. The structure of the model dictates that process instance trajectories linearly combine to produce the derivative trajectories. That requirement suggests that components and the target should share common high amplitude frequency peaks as well as low amplitude valleys. The data in this example does not seem to support this hypothesis because both relevant and irrelevant processes share similar amounts of peaks and valleys with the target. Unfortunately, Fourier analysis as a method to characterize process instances based on frequency amplitude peaks using this example data appears to have little value.

Another issue is that this analysis uses the magnitude of the harmonics as the primary means of characterizing the transformations. This may not be the most effective way of capturing a relationship between the Fourier transforms of the target derivative trajectories and the process instances. There could be an alternate way of analysing the results of the Fourier transform, such as considering the phase information more carefully, that could provide a better way to identify relationships between the process instances and the target derivative trajectory. In addition, the target derivative trajectories include coefficient values that will not be present in the process instance transformations. It is not clear how exactly the transformation accounts for that missing information in the process instances.

Other discouraging factors include the need to rely on only observed data, poor performance even with noise free data, and unknown scaling efficiency. Calculating the process instance trajectories relies on the assumption that all variables are observed and that the process instance equations contain only observed variables and parameters. If this restriction were relaxed in the future it would cause the analysis approach to fail. Noise will also make the results of this analysis worse. Noise present in the observations will be compounded when they are used to calculate the process instance trajectories. In addition, taking derivatives of the noisy observations tends to amplify noise. Finally, it is not known how well this technique will scale. It is promising because the Fourier transform will need to be performed once for each process instance trajectory as well as all derivatives so that aspect should scale linearly.

The selection algorithm in the best case would allow for a rank ordering of the process instances. These challenges ultimately lead to the Fourier analysis technique being abandoned in favour of a regression-based, backwards elimination heuristic.

7.2. A backwards elimination selection heuristic

Individual process instances are not predictive in the way that is traditionally true when using typical statistical regression predictors. Each term that appears in a regression equation generally accounts for a certain amount of the variance in the data regardless of what other terms appear with it in the equation. This happens when the data is independent and identically distributed (IID). This important assumption is violated in the time series data that is often used to construct process models. When regression is used on time series data, a single correct predictor does necessarily not produce a better score than a single incorrect predictor. This means that forward selection algorithm based on regression would not work. All of the relevant process instances must be present in order for the regression algorithm to return a high score. Fortunately, this remains true if irrelevant process instances are present in the equation as well. This suggests that a regression-based, backwards selection technique could work well.

Backwards elimination is a technique that starts with all terms present and removes one at a time until the fit score drops substantially. When applied as part of multivariate regression this method is ineffective for large numbers of terms due to round-off errors when predictor variables are highly collinear. It would cause instability in the algorithm and produce unreliable results. This situation occurs frequently during initial testing of this approach because many process instances can share variables and large numbers of terms can also be present.

In order to address these issues, a backwards elimination technique with repeated sampling was developed and implemented as part of a Selection-based Process Modeller (SPM). Instead of beginning with all possible process instances in an equation, it randomly selects a smaller subset of them. Using a subset of possible process instances, instead of all of them, avoids the problem of round off errors with large numbers of terms. SPM builds upon many of the advancements of the discovery system RPM which was discussed in chapter four. The selection algorithm still operates by searching through a space of process instance combinations one equation at a time to eventually find a complete model containing several equations that are able to explain the data. For each dependent variable, the equation-finding module in SPM repeatedly selects a subset of k processes, with uniform probability. Using the derivatives and rates from each process instances for each step, it then invokes multivariate regression to construct a linear equation that predicts the derivatives in terms of the process instance rates. If this equation's r^2 score is lower than a user-specified threshold, indicating that the all of the relevant process instances are not present in the equation, then SPM abandons it. Otherwise the system ranks the processes by their coefficients' absolute values and removes them in turn, smaller values first, rerunning the regression algorithm in each case. This continues until either the r^2 score falls below a given threshold or the ratio between new and old r^2 scores is less than a specified fraction. The result is a differential equation with only those processes responsible for its high predictive accuracy. It retains the equation for future use and repeats the procedure until the module has found a user-specified number of desired equations or examines a user specified maximum number of samples without reaching the desired number of acceptable equations. The program repeats these steps

separately for each dependent variable, removing any redundant equations that it reaches from different starting points.

7.3. Other features of SPM

SPM includes other features that improve its coverage and performance over RPM. One of these features includes a more flexible notation for generic processes that avoids the problem of over eager binding of variables to processes. SPM also includes a new method of constructing complete process models from individual equations. The new method still finds a set of differential equations separately for each dependent variable but then attempts to combine elements of each set into one or more consistent process models. It does this using a depth-first search that begins with the equations for one dependent variable at a time. Starting with the set of equations that were found to fit first variable well, it searches for all possible consistent process models for each equation. The search is not exhaustive, as it still requires that processes definitions be respected between equations. A process definition includes information about the variables that are influenced by that process. The model search proceeds one equation at a time so if a process appears in one equation the process definition can further require that the process also appear in other equations. These additional features help organize generic processes into related families that all share the same process rate equation, but could influence different variables.

For example, in chemistry the rate equation can be the product of the reactant species concentrations A and B. It is possible based on the generic process definitions that those two reactants A and B could produce chemical C, or both C and D. Observed data must be used to determine which one of those options is a better account of the observations. Each possibility is represented by a separate process instance that essentially states A + B -> C or A + B -> C + D. Both process instances would have the same rate expression but the first instance would appear in three equations, while the second instance would appear in four equations. Both process instances have the same rate expression so the regression algorithm will return a good fit for either instance when it appears the equations for A, B and C. The difference will arise in equation D. RPMs search strategy could pick the wrong version of the rate initially and it would be unable to correct its decision later. SPM is able to delay the decision about which version of that process instance to include in the model until it has evidence to support the decision.

In addition, SPM's sampling search method does not use knowledge from previous equations to constrain the search for later equations, as was the case with RPM and APM. Instead, each equation is constructed using sampling and backwards selection independently of every other equation. Often there will be more than one combination for a given equation that explains the data well. Once the system is done searching for each equation, it then attempts to combine the individual equations into a complete model using the process definitions to determine valid complete models. This provides the system with the ability to return multiple candidate models that can satisfactorily explain the data.

7.4. Experimental evaluation of the selection algorithm

The selection algorithm of SPM was designed to address the scaling issue faced by RPM. Whether it can accomplish this aim is an empirical question. This section presents two claims about the new system and reports evidence from experiments to support them. First, the selection algorithm in SPM does not

reduce the coverage compared to the previous system (RPM). Second, evidence is presented that shows the superior scaling performance of the SPM system.

To evaluate the claim of equivalent coverage, both systems were run on five different ecological time series data sets. These included natural data on protest interactions which first appeared during the initial development of RPM from chapter four, noise-free synthetic data for two predator-prey settings that involved six interacting organisms which were used during the development of APM in chapter six, and a new synthetic data set for an aquatic ecosystem involving zooplankton, phytoplankton, two nutrients, and detritus. For the natural data set, RPM and SPM performed identically in that they both found the same process model with the same fit score. For the synthetic domains both systems successfully found the target model used to generate the data, although in some cases SPM also found models with similar r^2 scores.

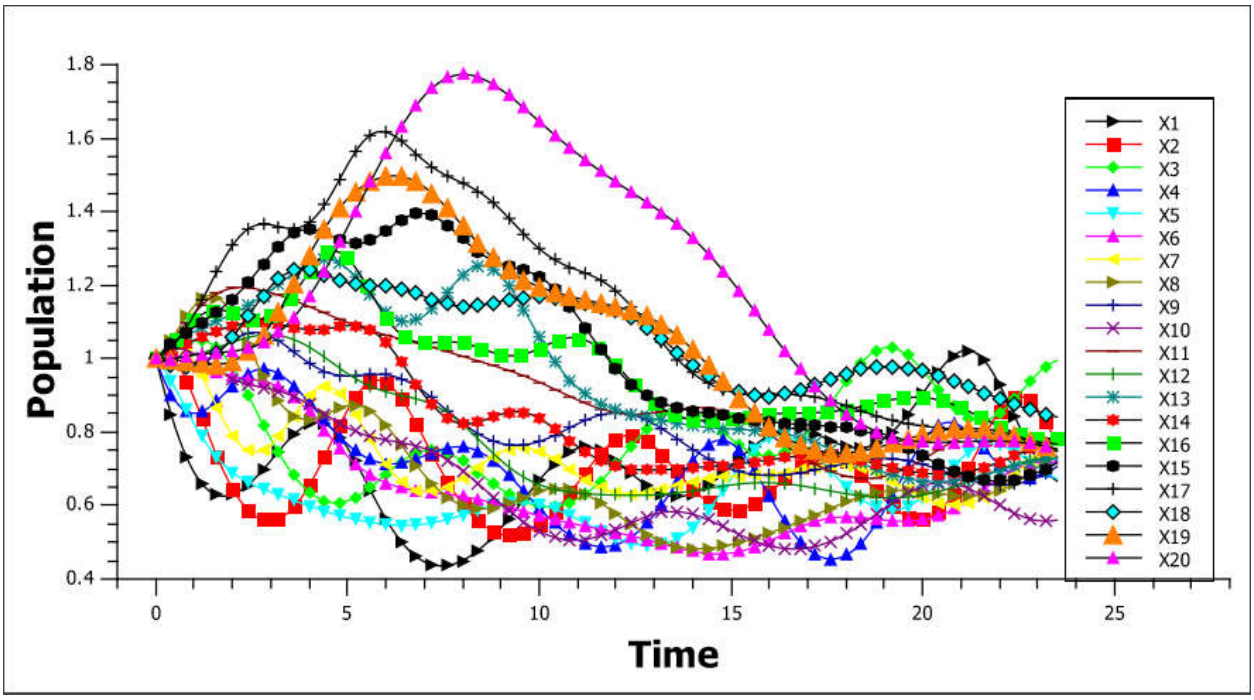


Figure 46 Observed trajectories for a 20-variable predator prey system. The large number of variables illustrates the necessity of using computational scientific discovery system to understand the dynamics.

The new system is also able to reconstruct a 20-variable predator prey model from the multivariate trajectories shown in Figure 46. For this task SPM considered differential equations with ten or fewer rate terms and sampled no more than 1,000 subsets of rate expressions. Over 50 runs, the system reliably found a single consistent model, which corresponded to the target, in 1.35 ± 1.28 CPU seconds, as compared to 0.38 ± .018 seconds for its predecessor. The new system requires more time in this case because it searched for 1,000 equations, which can require more time in less complex domains compared to RPMs greedy search strategy. SPM also reliably reconstructed a six-variable predator prey model when given not only two relevant generic processes, but also 16 irrelevant ones that specified different rate expressions which produced 324 distinct process instances. In this case the system

required 4,000 samples to find each equation and took 2.7 ± 1.2 seconds on average compared to RPM's 9.1 ± 0.2 seconds.

Table 10 Differential equations for a chemical system with six variables that interact through eight distinct reactions. SPM can reconstruct this model from the time series whereas RPM cannot.

$$\frac{dx_1}{dt} = 1.1x_1x_2 - 1.6x_1$$

$$\frac{dx_2}{dt} = 1.8\,x_1 - 1.5x_2 - 1.0\,x_2x_3 + 0.9\,x_5x_6$$

$$\frac{dx_3}{dt} = 1.9\,x_1 + 1.1\,x_2 - 1.3\,x_3 - 1.3\,x_2x_3$$

$$\frac{dx_4}{dt} = 0.9\,x_2 + 0.8\,x_3 - 2.5\,x_4x_5 + 0.5\,x_5x_6$$

$$\frac{dx_5}{dt} = 0.9\,x_3 - 1.\,x_4x_5 + 0.9\,Z$$

$$\frac{dx_6}{dt} = 2.3\,x_4x_5 - 0.8\,x_5x_6 - 0.5\,x_6$$

In addition, both systems were run on a number of synthetic chemical data sets. Table 10 shows the equations for one chemical system with six variables connected through eight reactions (processes). One process involved a time-varying influx variable *Z* that keeps the other variables from reaching a steady state. Trajectories for this system are shown in Figure 47. In another chemical data set, ten chemicals participated in 12 reactions, including a time-varying influx. Trajectories from this system are shown in Figure 48. SPM had no difficulty inducing both the reaction network from the given trajectories and the three required generic process definitions. In the first case, the system generated 22 process instances, then took 1000 samples of six rate terms to identify each component equation. In the second case, it generated 46 processes from four generic process definitions, then took 15,000 samples of ten rate terms. Runs on the first data set required 14.7 ± 0.21 CPU seconds on average whereas those for the second took a mean of 111.8 ± 0.6 seconds. In contrast, RPM generated 63 process instances from analogous generic process definitions and could not induce either target model. RPM failed to find any models in the chemical domain due to its previous discussed over eager binding of processes. These runs demonstrated that SPM is able to induce process models in the domain of chemistry that its predecessor could not handle.

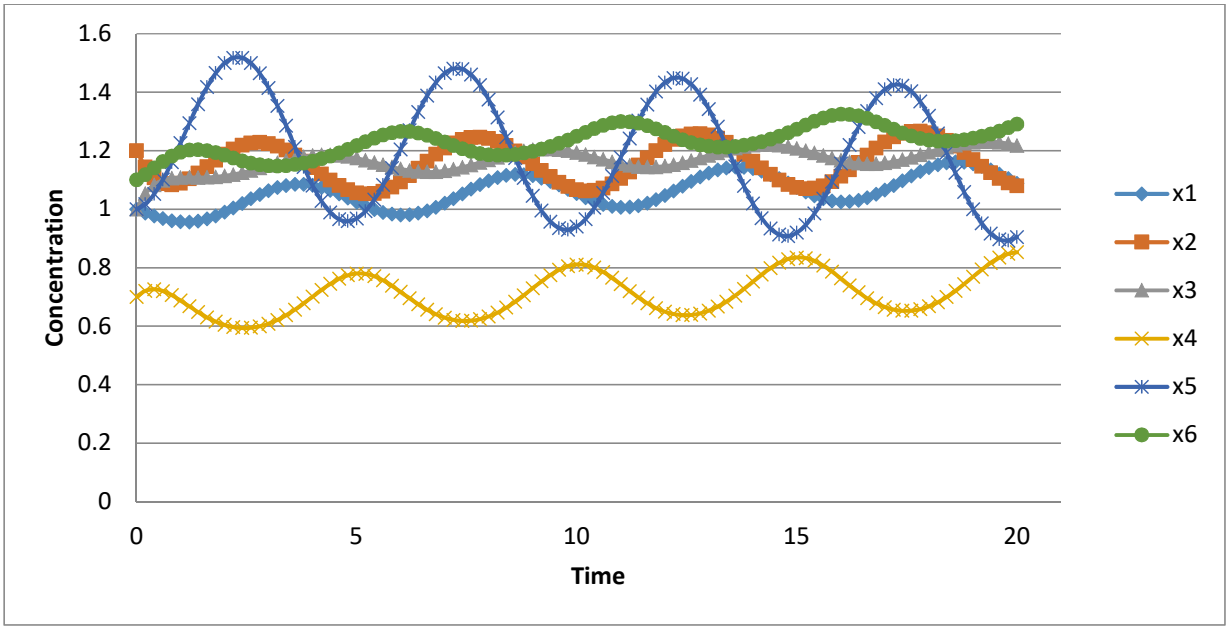


Figure 47 Trajectories from a chemical system involving six interacting chemicals with equations shown in Table 10

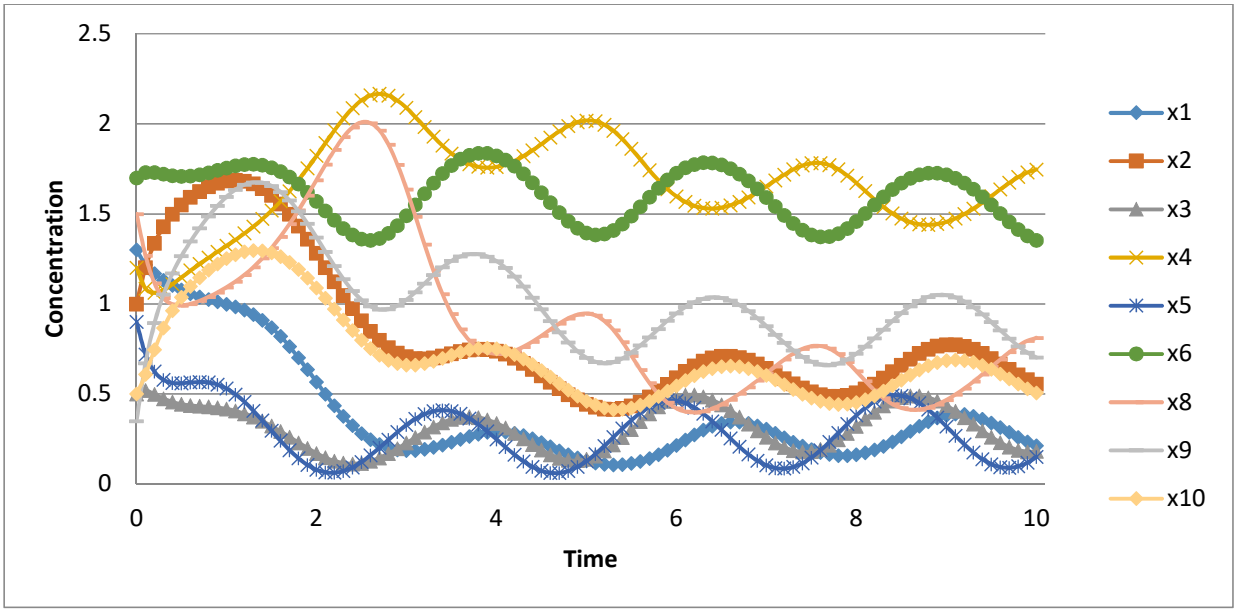


Figure 48 Trajectories for a chemical systems with ten variables interacting through 12 reactions.

7.5. Scalable induction of differential equations

The random sampling with backwards elimination should produce a search algorithm that scales better than the exhaustive search method used by the previous system. To test this assertion, synthetic data sets were generated in which derivatives were a linear function of different numbers, from one to ten, of processes with random valued rates. The random data ensure that the terms in the equation were not highly correlated, thus reducing redundant information. In this experiment it was not desirable to have SPM produce alternative models that explain the data. Each system was run ten times on each

equation and the CPU time to find the equation was measured. The number of samples for SPM was fixed at 10,000 and the number of terms in the equation was set to 13 for all runs.

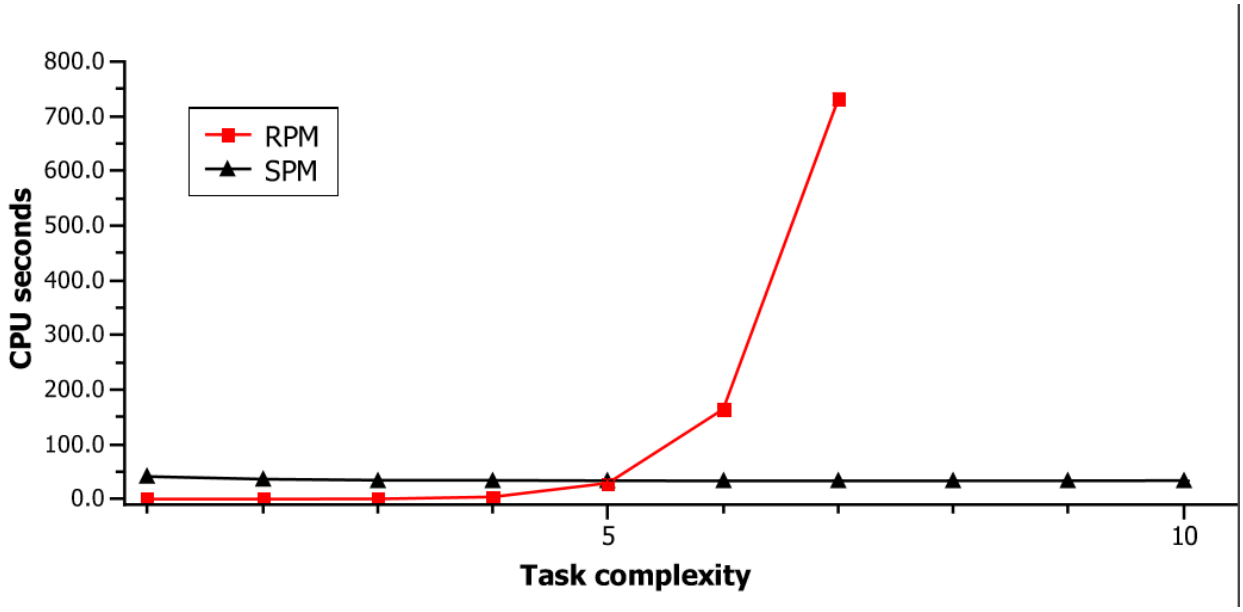


Figure 49 Average time for SPM and RPM to find target equations, in CPU seconds.

Figure 49 presents the results of the experiment. RPM actually finds simpler equations more rapidly than SPM, consistent with its simplicity bias. This changes for equations with five processes, at which point SPM becomes faster. In fact the combinations grew so quickly that RPM could not complete its search in a reasonable time once there were 9 terms in the equation. Growth in CPU time for SPM was essentially constant because the number of samples to search was the same in each case.

SPM's sampling approach does not guarantee that it will find the appropriate equation. The correct set of rates must appear in the set of sampled rates and the feature selection algorithm must be able to correctly identify them. The probability that the correct combination of rates will appear follows the same logic as hitting winning lottery numbers. The winning set of process rates must appear in the selected sample from the pool of all possible rates. Using *T* for the total number of processes in the pool and *R* as the number of processes in the target equation, the total number of ways that *T* processes can appear in *R* positions is calculated using $\binom{T}{R} = \frac{T!}{R!(T-R)!}$. The algorithm uses backwards selection and chooses as sample size that is expected to be larger than the winning set, leading to the equivalent lottery calculation of only needing to match some of the numbers instead of all of them, thus improving the odds. The odds of winning becomes the number of ways to select the winning process rates multiplied by the all the different ways that irrelevant rates can appear along with the winning rates. Using S as the size of the sample, the total number of equations that can include the correct set of process rates is $\binom{T}{S}\binom{T-S}{S-R}$. The probability of selecting an equation that includes the correct set of process rates can be calculated by dividing that by the total number of ways *T* processes can appear in *R* positions, resulting in the final probability calculation $\binom{T}{S}\binom{T-S}{S-R}/\binom{T}{R}$. Additional sampling increases the odds of the correct set of process instances appearing in an equation just as buying more tickets increases the odds of winning, but it also increases the CPU time further.

There are a number of areas where the selection algorithm can be improved. For one, it is fundamentally reliant on probability so there is no guarantee that it will be able to find a model that explains the data, if one exists. This is a natural trade-off between coverage of the search space and the time to search through it. The system also relies on many user-specified parameters including the total number of samples to take, the number of terms in the equation for backwards sampling, and the total number of equations to attempt to find for each dependent variable as well as the r^2 score. The use of synthetic data provided a way to determine acceptable values since the target model was known. The number of terms to sample as well as the total number of samples per equation should translate well to empirical data as these parameters are based on probabilistic assumptions that do not change with the data source. However the number of equations to find as well as the acceptable r^2 score will depend heavily on how many different accounts for the data can exist given the available process library and what the user deems an acceptable fit for the domain. SPM also shares the same assumption of previous systems that the process rates are determined by parameter-free algebraic expressions. This assumption allows for the use of multiple linear regression which guarantees that the coefficients that are returned are optimal. However, many domains can include rate expressions that contain unknown parameters and the current system would be unable to handle those.

8. Conclusion

Computational scientific discovery aims to provide scientists with the tools to help automate aspects of the discovery process in order to improve both the speed and accuracy of discoveries. The need has become more critical as the amount of available data has grown. Unlike the fields of data mining and machine learning, computational scientific discovery focuses on increasing the understanding of system behaviour, not just predicting it. This approach places an emphasis on incorporating formalisms from the domains being modelled to help facilitate explanation. The domain formalisms that are incorporated into the model help it to support communication, prediction, and experimentation. Process models represent one type of modelling paradigm that has applications in many domains. Processes define causal relationships between elements of systems and form the basis for explaining the behaviour of systems.

Process models as a computational scientific discovery tool were introduced in 2002 and have continued to develop. The current state of the art approach implemented in the discovery system SC-IPM has several important drawbacks that make it computationally intensive including the need to evaluate only complete model structures, the use of simulation during parameter estimation, and the non-linear nature of the parameter space. The work presented throughout this thesis is intended to primarily address the scalability concerns of the previous approach by developing a new framework for process models built around the concept of rate-based processes.

One chapter is dedicated to addressing each of the five primary aims outlined in the introductory chapter. The first aim was to document and demonstrate the performance and reliability of the existing discovery system SC-IPM and illustrate why it is necessary to develop a new approach rather than attempt to modify the existing system. The second aim involved providing a detailed description of the prototyping process of the new regression based parameter estimation algorithm in order to show that it has the potential to address many of the performance and reliability issues of the existing system. The third aim was to incorporate this new algorithm into a working discovery system and empirically demonstrate that the new system has equivalent coverage while performing much faster as measured by CPU time. The fourth aim involved demonstrating the model adaptation capabilities of the framework and providing evidence that adaptation provides significant CPU savings over building a model from scratch. The fifth and final aim was to address the poor scaling performance of the exhaustive search strategy.

A review of the literature in chapter two presented details of several model building techniques that can be used as part of computational scientific discovery tool. Each approach has different strengths and weaknesses that make it more or less applicable to the kinds of discoveries of interest. Many of the techniques of data mining and machine learning are flexible and highly automatic but the models they produce do not provide good explanations. The work in this thesis builds upon the work of qualitative process theory that uses a similar notion of processes to specify a causal relationship between variables but in this case processes define quantitative relationships.

Chapter three addresses the first aim by detailing the performance and reliability of SC-IPM, the system that this work builds upon, highlighting specific improvements that needed to be addressed. These include that only complete model structures can be evaluated, a simulation must be performed to

estimate parameters, and that the parameter estimation algorithm must be repeated several times to increase its reliability. These factors contribute to the overall slow and unreliable performance shown in Table 2 and Figure 22 from chapter three where SC-IPM often fails to find a set of parameters for a known model structure that fit the data well. In response, a new regression-based parameter estimation technique was introduced. That chapter also provides a detailed description of the prototyping process showing that it can address the shortcomings of the existing system, which is the second aim of the thesis. Evidence was presented to show that when the algorithm was given an appropriate model structure, the parameter estimates using a regression based algorithm are reasonable. In addition, the new approach can correctly reject erroneous model structures. The evidence was encouraging enough that a new process modelling framework was developed in order to incorporate the regression based parameter estimation algorithm into a new discovery system.

The third aim, to implement the new algorithm into a working discovery system, is addressed in chapter four when a new discovery system called the Regression-guided Process Modeller (RPM) is introduced. The chapter describes a number of important simplifying assumptions to help make the implementation feasible while still maintaining broad applicability. These include that all processes have a rate at which they operate, derivatives are influenced in a direct proportion to a process rate, all variables are observed, and that parameters in the rate equations are known. RPM, like its predecessor SC-IPM, still uses generic processes to encode knowledge and expresses models using processes to help facilitate explanation and understanding. Empirical evidence is presented that shows that RPM is able to find the same models that the previous system but does so over 80,000 times faster and that it is able to more reliably find models that explain the data well.

RPM is the first ever implementation of a rate-based process modelling framework. The various evaluations on the system provided evidence that this approach discovers models that explain data. The system demonstrated many important ideas but left several areas that can be improved or deserve additional exploration. One of the most significant is that the system incorporates a greedy search for models that does not backtrack and produces only a single model result. This issue is addressed to some degree in chapter seven with the auxiliary changes implemented into the Selection-based process modeller (SPM). Another important limitation of RPM is that the parameter estimation algorithm requires that the rate expression be parameter free. This allows the rate expression to be calculated and thus linear regression can be used to estimate the coefficients. The use of a non-linear regression technique will re-introduce the problem of local optima and reduce the reliability of parameter estimation. However, under the rate-based process framework each equation is treated independently, so the parameter identification problem is subdivided into independent equations. Each equation still does not require simulation to estimate the parameter values which can greatly reduce the computational effort of this step. In cases where the same parameter appears in multiple equations it is possible and necessary to re-use known parameter values from previous equations in later equations, further reducing the computational effort required. The separation of the model structure into independent equations provides a straightforward way to approach model adaptation.

Before getting into adaptation, chapter five introduced many important concepts related to synthetic data and model simulation. These supporting concepts are critical for the continued development and evaluation of computational scientific discovery systems. The primary advantage of synthetic data is that it provides a known target model during discovery so that the search algorithms can be verified. In realistic settings, the model that accounts for the data is unknown and there may be several alternative

models that can account for the data equally well. This can occur of the threshold for what qualifies as a good fit is low, such as when the available data contains a lot of noise. Simulation of data can also be a challenging problem. In most cases the model is expressed as a set of differential equations that do not have an analytical solution, so a numerical approximation must be used. This can introduce significant error if not performed carefully.

A new algorithm for model adaptation was implemented into a fully functional discovery system called Adaptive Process Modeller (APM) in chapter six and its performance was evaluated. APM address the fourth aim of the thesis, to demonstrate the adaptation capability of the rate-based process model framework. APM takes advantage of the independent nature of the model equations in order to perform a three stage revision procedure. It begins with anomaly detection in order to determine which equations, if any, need to be altered. It then tries to fit the new data by re-estimating the parameters and then resorting to structure modification as needed. Evidence was presented to show that revising an existing model was up to 87 times faster than building a model from scratch and that more benefit was seen when more of the existing model could be reused.

Adaptation introduces a number of directions that the system can be extended. One involves enhancing APM to operate on multivariate time series that arrive in a continuing stream. The system would monitor each variable for deviations from predicted values and invoke the revision process, possibly more than once, when the model appears to have sufficiently changed. Anomaly detection should be inexpensive and adaptation effort grows linearly with the number of equations changed. Rate-based process models make predictions directly about derivatives, so there is no need for simulation if one can estimate derivatives from observed values. However, it is important to take care that parameter revision is not overused, as frequently changing parameter values could compensate for structural changes that would otherwise go unnoticed.

APM's method for revising the coefficients and structure of equations also needs improvement. It is an extension of the RPM system so it suffers from the same drawback of retaining only the best-scoring equation for a given anomalous derivative. The greedy search strategy means that if the system can make decisions that cannot be changed should additional information become available during search. Future research should explore a form of beam search to retain a set of best-scoring equations, instead of only one, and then select one or more of the best combinations once they have all been examined. The system also needs to incorporate a similar strategy for deciding whether a revised equation is good enough; this should reduce the need for fine tuning of the r^2 ratio, which is currently somewhat sensitive.

Another area for additional research involves process invention, which should prove useful when APM's methods for reestimating coefficients and altering equation structure are unable to explain new observations. In such cases, it could postulate entirely new processes, each of which has an algebraic rate expression and a set of proportional derivatives. However, recall that any variable which appears in the rate expression must also appear as a derivative and that, during revision, only anomalous derivatives can appear as influences. This should reduce the set of candidate processes substantially, making the discovery of new processes far more tractable in the context of revising models than when inducing them from scratch.

Despite the advances introduced by RPM and APM, neither system has any ability to select processes ahead of time. They rely on an exhaustive search method. The equations are organized into processes that allow earlier equations to constrain later ones which helps make the exhaustive search technique more tractable. However, even that will not allow these systems to scale to models that includes hundreds of variables. Another important limitation is that neither system can return more than one model during search.

Chapter seven introduced a selection-based process modeller (SPM) that was designed to address these concerns. This new algorithm satisfies the fifth and final aim of the thesis which is to incorporate a more scalable search strategy than the exhaustive method used in the first system. The chapter discusses a forward selection algorithm based on Fourier analysis that proved to be unsuccessful. However, a backwards selection algorithm based on regression was workable and was then used as the selection algorithm. It works by randomly selecting a subset of *n* process instances to appear in an equation then regression is used to find coefficients that best fit the estimated derivatives. If the equation has a fit score that is above a user-defined threshold, then the algorithm drops the term with the smallest absolute coefficient and repeat the fitting procedure with the remaining terms. This process repeats until the dropping of the lowest term results in a substantial drop in fit score, indicating that all of the terms in the equation are relevant. This approach proved to be effective and the evidence showed that the backwards selection algorithm complexity grew more slowly than RPM.

SPM also introduces the ability to return more than one candidate model. Search is undertaken for each equation individually and all equations that produce an acceptable fit are stored. This search can result in multiple different candidate equations per variable. When the search for individual equations is finished the system combines them into complete models in all ways that are consistent with the process definitions. This can result in many different models that provide an acceptable account of the data and is in contrast to the technique used by RPM and APM where the algorithm would search for an equation for a variable, then choose the best of the acceptable equations it found, then require that later equations be consistent with its current choice.

There are several areas where the selection algorithm can be improved. The algorithm requires the user to specify several search parameters and there is currently no guidance on what values are appropriate. The selection technique relies on sampling, so there is a natural trade-off between coverage of the search space and the time to search through it but there is no guarantee that the system will find a solution, if one exists. SPM also shares the same assumption as RPM and APM that process rates are determined by parameter-free algebraic expressions. This assumption simplifies the regression algorithm and guarantees that the parameters found are optimal but limits the types of equations the systems can handle. This is a limitation of these implementations and is not an assumption required as part of the rate-based process modelling framework.

The literature review discussed an equation discovery technique that leveraged the sparseness of the equation structure to make the identification problem tractable (Brunton et al., 2016). One of their techniques included incorporating the least absolute shrinkage and selection operator (Tibshirani, 1996). This same regression operator can likely be used as part of a selection heuristic as the search space of process instances also qualifies as sparse in many cases. This could prove to be a more effective selection method that would further improve the system performance.

Another area of future investigation includes classifying the rate terms using qualitative or quantitative metrics. The extra effort in classification could pay off for very large spaces. SPM's search strategy involves taking multiple samples of rate terms. As SPM takes more samples it could learn which rate terms are most likely to be irrelevant. The entire class of rate terms can then be excluded from search or otherwise de-emphasized, which should improve the odds that a promising combination of rate terms will be found. The classification could be as simple as tabulating the probability that a particular term is one of the first to be dropped from an equation. Often, however, the rate terms are highly collinear so classification may prove difficult.

There are a number of other research directions for the rate-based process model framework. One involves the ability to handle missing or hidden variables. In many cases not all variables are directly observable or may have not been measured. In certain conditions it is possible to identify a single hidden variable. It may be possible to expand this idea to handle multiple hidden variables. A forward selection algorithm would be another useful area for further research. This thesis presented one attempt to develop a forward selection algorithm that used Fourier analysis but that approached proved to be ineffective. However, a different approach using other techniques may be able to successfully characterize processes instance trajectories so that the more promising ones can be identified prior to inserting them into a model.

Another important avenue for the development of rate-based process models includes various kinds of ease of use improvements for the end-user. One of the aims of computational scientific discovery is to create tools that assist scientists in making discoveries. To this end, this entire discipline may benefit from many user-centred design concepts. The tools should be designed with the scientists who will be using them in mind. For example, these systems should be developed using common, free, and accessible languages with large support communities. This will make it easier to find people with the ability to modify and further develop the tools. Along these lines, the final software artefact should be easy to use and have good documentation and several example empirical and synthetic data sets with associated models as well as process libraries, all from multiple domains.

Computational scientific discovery draws a wide variety of techniques to accomplish its ambitious goal of automating scientific discovery. The introduction of a rate-based process modelling framework provides important contributions to this field. The various implementations have demonstrated improved performance over previous systems in both domain coverage and computational time. These improvements allow for discoveries to be made in a wider variety of domains and the models that are discovered can explain a greater degree of complexity in the data. There are several directions where the rate-based process modelling framework can be expanded so that it is an even more effective tool for computational scientific discovery. The future of using rate-based processes for computational scientific discovery is only just beginning and the work presented in this thesis represents the first steps in this area of research.

References

Ankerst, M., Breunig, M. M., Kriegel, H., & Sander, J. (1999). OPTICS: Ordering points to identify the clustering structure. *ACM Sigmod Record, 28*(2) 49-60.

Arvay, A., & Langley, P. (2015). Heuristic adaptation of rate-based process models. *Proceedings of the Third Annual Conference on Advances in Cognitive Systems ACS, 1* 17.

Ascher, U. M., & Petzold, L. R. (1998). *Computer methods for ordinary differential equations and differential-algebraic equations* Siam.

Asgharbeygi, N., Langley, P., Bay, S., & Arrigo, K. (2006). Inductive revision of quantitative process models. *Ecological Modelling, 194*(1), 70-79.

Bay, S. D., Shrager, J., Pohorille, A., & Langley, P. (2002). Revising regulatory networks: From expression data to linear causal models. *Journal of Biomedical Informatics, 35*, 289-297. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.4129

Berkhin, P. (2006). A survey of clustering data mining techniques. *Grouping multidimensional data* (pp. 25-71) Springer.

Bifet, A., & Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. *Proceedings of the 2007 SIAM International Conference on Data Mining,* 443-448.

Bloomfield, P. (2004). *Fourier analysis of time series: An introduction* John Wiley & Sons.

Bobrow, D. G. (1984). Qualitative reasoning about physical systems: An introduction. *Artificial Intelligence, 24*(1), 1-5. 10.1016/0004-3702(84)90036-5 Retrieved from http://www.sciencedirect.com/science/article/pii/0004370284900365

Borrett, S. R., Bridewell, W., Langley, P., & Arrigo, K. R. (2007). A method for representing and developing process models. *Ecological Complexity, 4*(1–2), 1-12. 10.1016/j.ecocom.2007.02.017 Retrieved from http://www.sciencedirect.com/science/article/pii/S1476945X07000177

Bradley, E., Easley, M., & Stolle, R. (2001). Reasoning about nonlinear system identification. *Artificial Intelligence, 133*(1-2), 139-188.

Brailsford, S. C. (December 2008). System dynamics: What's in it for healthcare simulation modelers. 1478-1483. 10.1109/WSC.2008.4736227

Bridewell, W., & Langley, P. (2010). Two kinds of knowledge in scientific discovery. *Topics in Cognitive Science, 2*(1), 36–52. 10.1111/j.1756-8765.2009.01050.x Retrieved from http://onlinelibrary.wiley.com/doi/10.1111/j.1756-8765.2009.01050.x/abstract

Bridewell, W., Langley, P., Racunas, S., & Borrett, S. (2006). Learning process models with missing data. *European Conference on Machine Learning,* 557-565.

Bridewell, W., & Todorovski, L. (2007). Learning declarative bias. *International Conference on Inductive Logic Programming,* 63-77.

Bridewell, Langley, Todorovski, & Džeroski. (2007). Inductive process modeling. *Machine Learning, 71*(1), 1-32. 10.1007/s10994-007-5042-6 Retrieved from http://link.springer.com/10.1007/s10994-007-5042-6http://www.springerlink.com/index/pdf/10.1007/s10994-007-5042-6

Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences, 113*(15), 3932-3937. 10.1073/pnas.1517384113 Retrieved from http://www.pnas.org/content/113/15/3932.abstract

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery, 2*(2), 121-167.

Chartrand, R. (2011). Numerical differentiation of noisy, nonsmooth data. *ISRN Applied Mathematics, 2011*

Cheeseman, P., Self, M., Kelly, J., Taylor, W., Freeman, D., & Stutz, J. C. (1988). Bayesian classification. *Aaai, 88* 607-611.

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association, 74*(368), 829-836.

Cornforth, T. W., & Lipson, H. (2013). Inference of hidden variables in systems of differential equations with genetic programming. *Genetic Programming and Evolvable Machines, 14*(2), 155-190. 10.1007/s10710-012-9175-4 Retrieved from https://link.springer.com/article/10.1007/s10710-012-9175-4

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273-297. 10.1007/BF00994018

Darden, L. (1990). Diagnosing and fixing faults in theories. In J. Shrager, & P. Langley (Eds.), *Computational models of scientific discovery and theory formation* (pp. 319-354) Morgan Kaufmann.

Datta, B. N. (2010). *Numerical linear algebra and applications, second edition* (2nd ed.). Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Draper, N. R., Smith, H., & Pownell, E. (1966). *Applied regression analysis* Wiley New York.

Dunham, M. H. (2003). *Data mining introductory and advanced topics*. Upper Saddle River, NJ: Upper Saddle River, NJ : Prentice Hall/Pearson Education 2003.

Džeroski, S., Langley, P., & Todorovski, L. (2007). Computational discovery of scientific knowledge. In S. Džeroski, & L. Todorovski (Eds.), *Computational discovery of communicable scientific knowledge* ()

Džeroski, S., & Todorovski, L. (2008). Equation discovery for systems biology: Finding the structure and dynamics of biological networks from time course data. *Current Opinion in Biotechnology, 19*(4), 360-368. 10.1016/j.copbio.2008.07.002 Retrieved from http://www.sciencedirect.com/science/article/pii/S0958166908000839

Easley, M., & Bradley, E. (2007). Incorporating engineering formalisms into automated model builders. In S. Džeroski, & L. Todorovski (Eds.), *Computational discovery of scientific knowledge* (pp. 44–68). Berlin, Heidelberg: Springer-Verlag. Retrieved from http://dx.doi.org/10.1007/978-3-540-73920-3_3

Ester, M., Kriegel, H., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Kdd, 96*(34) 226-231.

Fahlman, S. E. (2012). Beyond idiot-savant AI. *Advances in Cognitive Systems, 1*(1) Retrieved from http://cogsys.org/journal/volume-1

Falkenhainer, B. (1990). A unified approach to explanation and theory formation. In J. Shrager, & P. Langley (Eds.), *Computational models of scientific discovery and theory formation* (pp. 157-196) Morgan Kaufmann.

Falkenhainer, B., & Forbus, K. D. (1991). Compositional modeling: Finding the right model for the job. *Artificial Intelligence, 51*(1), 95-143. 10.1016/0004-3702(91)90109-W Retrieved from http://www.sciencedirect.com/science/article/pii/000437029190109W

Fekih, A., Xu, H., & Chowdhury, F. N. (2007). Neural networks based system identification techniques for model based fault detection of nonlinear systems. *International Journal of Innovative Computing, Information and Control, 3*(5), 1073-1085.

Forbus, K. D. (1984). Qualitative process theory. *Artificial Intelligence, 24*(1–3), 85-168. 10.1016/0004-3702(84)90038-9 Retrieved from http://www.sciencedirect.com/science/article/pii/0004370284900389

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning, 29*(2-3), 131-163. 1007465528199

Friedman, S., & Forbus, K. (2010). An integrated systems approach to explanation-based conceptual change. *Twenty-Fourth AAAI Conference on Artificial Intelligence*

Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review, 13*(1), 3-54.

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. *Brazilian Symposium on Artificial Intelligence,* 286-295.

Gama, J., Žliobaite, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Comput. Surv., 46*(4), 44:1–44:37. 10.1145/2523813 Retrieved from http://doi.acm.org/10.1145/2523813

Garrett, S. M., Coghill, G. M., Srinivasan, A., & King, R. D. (2007). Learning qualitative models of physical and biological systems. *Computational discovery of scientific knowledge* (pp. 248-272) Springer.

Gorunescu, F. (2011). In SpringerLink (Online service) (Ed.), *Data mining concepts, models and techniques*. Berlin ; Heidelberg: Berlin ; Heidelberg : Springer c2011.

Grauer, J. A., & Hubbard Jr, J. E. (2013). *Flight dynamics and system identification for modern feedback control: Avian-inspired robots* Elsevier.

Hand, D. J., Mannila, H., & Smyth, P. (2001). *Principles of data mining* MIT press.

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society.Series C (Applied Statistics), 28*(1), 100-108.

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America-Biological Sciences, 79*(8), 2554-2558. 10.1073/pnas.79.8.2554

Hunt, J. E., Lee, M. H., & Price, C. J. (1993). Applications of qualitative model-based reasoning. *Control Engineering Practice, 1*(2), 253-266. 10.1016/0967-0661(93)91615-4 Retrieved from http://www.sciencedirect.com/science/article/pii/0967066193916154

Iba, H. (2008). Inference of differential equation models by genetic programming. *Information Sciences, 178*(23), 4453-4468. 10.1016/j.ins.2008.07.029 Retrieved from http://www.sciencedirect.com/science/article/pii/S0020025508002909

Ivanovska, A., Todorovski, L., Debeljak, M., & Džeroski, S. (2009). Modelling the outcrossing between genetically modified and conventional maize with equation discovery. *Ecological Modelling, 220*(8), 1063-1072. 10.1016/j.ecolmodel.2009.01.035 Retrieved from http://www.sciencedirect.com/science/article/pii/S0304380009000957

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *Acm Computing Surveys, 31*(3), 264-323. 10.1145/331499.331504

Jensen, F. V. (1996). *An introduction to bayesian networks* UCL press London.

Kaufman, L., & Rousseeuw, P. J. (1990). Partitioning around medoids (program pam). *Finding Groups in Data: An Introduction to Cluster Analysis,* 68-125.

Khan, S., Yufeng, L., & Ahmad, A. (2009). Analysing complex behaviour of hydrological systems through a system dynamics approach. *Environmental Modelling & Software, 24*(12), 1363-1372. 10.1016/j.envsoft.2007.06.006 Retrieved from http://www.sciencedirect.com/science/article/pii/S1364815207001235

King, R. D., Garrett, S. M., & Coghill, G. M. (2005). On the use of qualitative reasoning to simulate and identify metabolic pathways. *Bioinformatics, 21*(9), 2017-2026. 10.1093/bioinformatics/bti255

Retrieved from https://academic.oup.com/bioinformatics/article/21/9/2017/408960/On-the-use-of-qualitative-reasoning-to-simulate

Kokar, M. M. (1986). Determining arguments of invariant functional descriptions. *Machine Learning, 1*(4), 403-422.

Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. (2006). Machine learning: A review of classification and combining techniques. *Artificial Intelligence Review, 26*(3), 159-190. 10.1007/s10462-007-9052-3 Retrieved from https://link.springer.com/article/10.1007/s10462-007-9052-3

Kuipers, B. (1986). Qualitative simulation. *Artificial Intelligence, 29*(3), 289-338. 10.1016/0004-3702(86)90073-1 Retrieved from http://www.sciencedirect.com/science/article/pii/0004370286900731

Langley, P., & Arvay, A. (2015). Heuristic induction of rate-based process models. *Aaai,* 537-543.

Langley, P., Sanchez, J. N., Todorovski, L., & Dzeroski, S. (2002). Inducing process models from continuous data.

Langley, P., Shiran, O., Shrager, J., Todorovski, L., & Pohorille, A. (2006). Constructing explanatory process models from biological data and knowledge. *Artificial Intelligence in Medicine, 37*(3), 191-201.

Langley, P., Simon, H. A., & Bradshaw, G. L. (1987). Heuristics for empirical discovery. *Computational models of learning* (pp. 21-54) Springer.

Langley, P., Simon, H. A., Bradshaw, G. L., & Zytkow, J. M. (1987). *Scientific discovery: Computational explorations of the creative process*. Cambridge, MA, USA: MIT Press.

Ljung, L. (2010). Perspectives on system identification. *Annual Reviews in Control, 34*(1), 1-12. 10.1016/j.arcontrol.2009.12.001 Retrieved from http://www.sciencedirect.com/science/article/pii/S1367578810000027

Lubansky, A. S., Yeow, Y. L., Leong, Y., Wickramasinghe, S. R., & Han, B. (2006). A general method of computing the derivative of experimental data. *AIChE Journal, 52*(1), 323-332.

Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (1983). *Machine learning: An artificial intelligence approach, volume I* Morgan Kaufmann.//doi.org/10.1016/C2009-0-27563-5

Monod, J. (1949). The growth of bacterial cultures. *Annual Reviews in Microbiology, 3*(1), 371-394.

Mooney, R. J., & Richards, B. L. (1992). Automated debugging of logic programs via theory revision. *Proceedings of the Second International Workshop on Inductive Logic Programming,* Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.147.3923

Murthy, S. K. (1998). Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery, 2*(4), 345-389.

Nayak, P. P., & Joskowicz, L. (1996). Efficient compositional modeling for generating causal explanations. *Artificial Intelligence, 83*(2), 193-227. 10.1016/0004-3702(95)00024-0 Retrieved from http://www.sciencedirect.com/science/article/pii/0004370295000240

O'Rorke, P., Morris, S., & Schulenburg, D. (1990). Theory formation by abduction: A case study based on the chemical revolution. In J. Shrager, & P. Langley (Eds.), *Computational models of scientific discovery and theory formation* (pp. 197-224) Morgan Kaufmann.

Pazzani, M. J., Mani, S., & Shankle, W. R. (2001). Acceptance of rules generated by machine learning among medical experts. *Methods of Information in Medicine, 40*(5), 380-385.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*(1), 81-106. 1022643204877 Retrieved from https://doi.org/10.1023/A:1022643204877

Rajamoney, S. (1990). A unified approach to empirical discovery. In J. Shrager, & P. Langley (Eds.), *Computational models of scientific discovery and theory formation* (pp. 225-254) Morgan Kaufmann.

Richardson, G. P., & Pugh, A. L. (1981). *Introduction to system dynamics modeling with dynamo*. Cambridge, MA, USA: MIT Press.

Saito, K., & Langley, P. (2007). Quantitative revision of scientific models. *Computational discovery of scientific knowledge* (pp. 120-137) Springer.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks, 61*, 85-117. 10.1016/j.neunet.2014.09.003 Retrieved from https://www-sciencedirect-com.ezproxy.auckland.ac.nz/science/article/pii/S0893608014002135

Schmidt, M., & Lipson, H. (2009). Distilling free-form natural laws from experimental data. *Science, 324*(5923), 81-85.

Simon, H. A. (1973). The structure of ill structured problems. *Artificial Intelligence, 4*(3-4), 181-201.

Simon, H. A., & Newell, A. (1958). Heuristic problem solving: The next advance in operations research. *Operations Research, 6*(1), 1-10.

Stolle, R., & Bradley, E. (2007). Communicable knowledge in automated system identification. *Computational discovery of scientific knowledge* (pp. 17-43) Springer, Berlin, Heidelberg. Retrieved from https://link.springer.com/chapter/10.1007/978-3-540-73920-3_2

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society.Series B (Methodological), 58*(1), 267-288. Retrieved from http://www.jstor.org/stable/2346178

Todorovski, L., Bridewell, W., Shiran, O., & Langley, P. (2005). Inducing hierarchical process models in dynamic domains. *Proceedings of the Twentieth National Conference on Artificial Intelligence,* 892-897. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.6347

Todorovski, L., & Dzeroski, S. (1997). Declarative bias in equation discovery. *Icml,* 376-384.

Todorovski, L., Džeroski, S., & Kompare, B. (1998). Modelling and prediction of phytoplankton growth with equation discovery. *Ecological Modelling, 113*(1–3), 71-81. 10.1016/S0304-3800(98)00135-5 Retrieved from http://www.sciencedirect.com/science/article/pii/S0304380098001355

Todorovski, L., Džeroski, S., Langley, P., & Potter, C. (2003). Using equation discovery to revise an earth ecosystem model of the carbon net production. *Ecological Modelling, 170*(2), 141-154.

Todorovski, L., Ljubič, P., & Džeroski, S. (2004). Inducing polynomial equations for regression. 441-452. 10.1007/978-3-540-30115-8_41 Retrieved from https://link.springer.com/chapter/10.1007/978-3-540-30115-8_41

Towell, G. G., Shavlik, J. W., & Noordewier, M. O. (1990). Refinement of approximate domain theories by knowledge-based neural networks. *Proceedings of the Eight National Conference on Artificial Intelligence,* 861-866. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.54.4913

Veilleux, B. G. (1979). An analysis of the predatory interaction between paramecium and didinium. *The Journal of Animal Ecology,* 787-803.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data mining: Practical machine learning tools and techniques* Morgan Kaufmann.

Zhang, G. P. (2000). Neural networks for classification: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 30*(4), 451-462.