November 2016

Automatic Development and Adaptation of Concise Nonlinear Models for System Identification

William G. La Cava
University of Massachusetts Amherst

AUTOMATIC DEVELOPMENT AND ADAPTATION OF CONCISE NONLINEAR MODELS FOR SYSTEM IDENTIFICATION

A Dissertation Presented

by

WILLIAM G. LA CAVA

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2016

Mechanical and Industrial Engineering

AUTOMATIC DEVELOPMENT AND ADAPTATION OF CONCISE NONLINEAR MODELS FOR SYSTEM IDENTIFICATION

A Dissertation Presented

by

WILLIAM G. LA CAVA

Approved as to style and content by:

Kourosh Danai, Chair

Lee Spector, Member

Matthew Lackner, Member

Sundar Krishnamurty, Department Chair
Mechanical and Industrial Engineering

ACKNOWLEDGMENTS

I would like to thank the following people for contributing to the work in this dissertation: Kourosh Danai; Lee Spector; Kushal Sahare; Matthew Lackner; Thomas Helmuth; Nic McPhee; Paul Fleming; Alan Wright; Blake Massey; Yahya Modarres-Sadeghi; and Banafsheh Seyed-Aghazadeh. In particular I would like to thank my advisor Kourosh for his selflessness and dedication to his research and his students. I would like to thank Lee for his enthusiastic engagement and inspiring breadth of interests. I would like to thank Matt for his support and guidance with wind turbine applications.

I also want to thank all of the friends and colleagues who kept me in good spirits in and out of the laboratory. I am incredibly grateful for the support of my partner Nelle Ward, my five siblings, and my loving parents Mary Lou and George.

Finally, I would like to dedicate this dissertation to my high school teacher Brian Spotts, who not only introduced me to programming in C and Java, but also taught me to type. He has forever influenced my career path.

This work is partially supported by the NSF-sponsored IGERT: Offshore Wind Energy Engineering, Environmental Science, and Policy (Grant Number 1068864). This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1053575 [202].

ABSTRACT

AUTOMATIC DEVELOPMENT AND ADAPTATION OF CONCISE NONLINEAR MODELS FOR SYSTEM IDENTIFICATION

SEPTEMBER 2016

WILLIAM G. LA CAVA

B.S., CORNELL UNIVERSITY

M.Eng., CORNELL UNIVERSITY

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Kourosh Danai

Mathematical descriptions of natural and man-made processes are the bedrock of science, used by humans to understand, estimate, predict and control the natural and built world around them. The goal of system identification is to enable the inference of mathematical descriptions of the true behavior and dynamics of processes from their measured observations. The crux of this task is the identification of the dynamic model form (topology) in addition to its parameters. Model structures must be concise to offer insight to the user about the process in question. To that end, this dissertation proposes three methods to improve the ability of system identification to identify succinct nonlinear model structures.

The first is a model structure adaptation method (MSAM) that modifies first principles models to increase their predictive ability while maintaining intelligibility. Model structure identification is achieved by this method despite the presence of parametric error through a novel means of estimating the gradient of model structure perturbations. I demonstrate MSAM's ability to identify underlying nonlinear dynamic models starting from linear models in the presence of parametric uncertainty. The main contribution of this method is the ability

to adapt the structure of existing models of processes such that they more closely match the process observations.

The second method, known as epigenetic linear genetic programming (ELGP), conducts symbolic regression without *a priori* knowledge of the form of the model or its parameters. ELGP incorporates a layer of genetic regulation into genetic programming (GP) and adapts it by local search to tune the resultant model structures for accuracy and conciseness. The introduction of epigenetics is made simple by the use of a stack-based program representation. This method, tested on hundreds of dynamics problems, demonstrates the ability of epigenetic local search to improve GP by producing simpler and more accurate models.

The third method relies on a multidimensional GP approach (M4GP) for solving multiclass classification problems. The proposed method uses stack-based GP to conduct nonlinear feature transformations to optimize the clustering of data according to their classes. In comparison to several state-of-the-art methods, M4GP is able to classify test data better on several real-world problems. The main contribution of M4GP is its demonstrated ability to combine the strengths of GP (e.g. nonlinear feature transformations and feature selection) with the strengths of distance-based classification.

MSAM, ELGP and M4GP improve the identification of succinct nonlinear model structures for continuous dynamic processes with starting models, continuous dynamic processes without starting models, and multiclass dynamic processes without starting models, respectively. A considerable portion of this dissertation is devoted to the application of these methods to these three classes of real-world dynamic modeling problems. MSAM is applied to the restructuring of controllers to improve the closed-loop system response of nonlinear plants. ELGP is used to identify the closed-loop dynamics of an industrial scale wind turbine and to define a reduced-order model of fluid-structure interaction. Lastly, M4GP is used to identify a dynamic behavioral model of bald eagles from collected data. The methods are analyzed alongside many other state-of-the-art system identification methods in the context of model accuracy and conciseness.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Scientific advancements depend on the ability of human experts to understand and mathematically describe the behavior of the dynamic systems they study. To this end, the field of system identification is mandated with inferring mathematical descriptions of physical processes using observations from those processes [121]. Since there are essentially infinite possible mathematical descriptions of the observable world, system identification must focus on the development of models that are most useful for engineers and scientists. Therefore, the goals of system identification extend beyond accurately representing the desired process: the model needs to cater to diverse scientific needs, such as control design or the human understanding of scientific discoveries. At its core, system identification is the approach whereby first principles and natural laws enter the vocabulary of the scientist: Newton's laws, for example, were deduced from his 17th century empirical observations [148]. This dissertation presents methods that automate the construction of intelligible and physically meaningful dynamic models from observations.

Traditionally, process dynamics are characterized by differential equations that are formulated by experts using first principles and/or empirical observations. These expert-designed differential equations then form the dynamic models that are used to estimate/predict process behavior. However, first-principles models often cannot fully characterize the nonlinear dynamics of the process, as represented by process observations. As a result, first-principles models may be abandoned in favor of empirical models such as linear and nonlinear autoregressive moving average (ARMAX) models [121, 12], neural networks [147, 53], and others [150, 141, 59, 19, 171], that utilize high-capacity structures to accommodate adaptation according to the measured process observations. Although these empirical models provide an

effective basis for estimation/prediction, they have two major drawbacks. The first is their 'black box' format, stemming from the imposed model structures, which obscures the acquired knowledge of the process. The second is their non-generalizability which makes them potentially deficient in representing the process under conditions (inputs) not encompassed by the measured observations.

To remedy the black box nature of these empirical models, models can be defined in symbolic form by symbolic regression [52, 21, 15] wherein both the structure (topology) and parameters (constants) are inferred from measured observations without presuming a model structure. These symbolic models have the potential to be intelligible and thereby allow users to understand the dynamics of the process for which they are developed. Symbolic regression is typically conducted using genetic programming (GP) [93], which is a bio-inspired machine learning technique that constructs a population of candidate models from mathematical building blocks and then selects and varies these models over several generations before converging on a model that best fits the process observations. Due to an expanded search space that includes model structures in addition to parameters, symbolic regression can be computationally intensive in comparison to other empirical methods. In addition, it can be difficult to leverage expert knowledge (e.g., a starting model) with the symbolic regression approach [179], although various developmental approaches have been proposed [95].

The contributions of this dissertation relate to improving the identification of model structures for system identification. Model structure identification is addressed in three scenarios. First, I consider the case in which a starting model (e.g., a first principles model) is available for representing the process (Ch. 2). A method for model structure adaptation (MSAM) is presented that optimizes nonlinear variable couplings in the neighborhood of the starting model using a gradient-based approach. Ch. 3 presents a method of inferring continuous dynamic models from observations without a starting model. In this case, an extension to traditional GP, known as epigenetic linear genetic programming (ELGP), is proposed that specifies local search of model structures using a representation and learning algorithm inspired by epigenetic processes in biology. Finally, Ch. 4 proposes a model struc-

ture identification method for multiclass classification known as M4GP, that again makes no a priori model assumptions. This method optimizes model structures as a set of feature transformations that project the measured observations into a more easily classified space.

In the second part of this dissertation, the methods presented earlier, i.e. MSAM, ELGP, and M4GP, are used to solve real-world dynamic modeling problems. Ch. 5 applies MSAM to the restructuring of controllers to account for plant nonlinearities. Ch. 6 applies ELGP to the identification of closed-loop wind turbine dynamics. Ch. 7 applies M4GP to the identification of agent-based models of bald eagle behavior. Finally, in Ch. 8, ELGP is applied to the identification of a dynamic model of vortex-induced vibration. New methods are presented for consolidating local models to represent global process physics.

The rest of this chapter is dedicated to formulating the system identification problem in the context of the proposed methods and summarizing broadly the literature on model structure identification, with a focus on GP-based methods. Due to the fairly unique contexts of the three methods presented in Ch. 2 - 4, I have decided to include separate Related Work sections in those chapters and to summarize the shared background of those methods briefly here.

1.1 System Identification

The underlying assumption of system identification is that there exists an analytical model of the system that would generate the measured observations $y(t_k)$ at the sample times $t_k = t_1, \ldots, t_N$ under the input, $\mathbf{u}(t)$, as

$$y(t_k, \mathbf{u}) = \hat{y}(t_k, M^*(\mathbf{x}, \mathbf{u}, \boldsymbol{\Theta}^*)) + \nu; \qquad k = 1, \ldots, N \tag{1.1}$$

where $\hat{y}$ is the model output, ν represents measurement noise in y, $\mathbf{x} = [x_1, \ldots, x_n]^T$ is the vector of state variables, and $M^*(\mathbf{x}, \mathbf{u}, \boldsymbol{\Theta}^*)$ denotes the correct model form embodied by the correct parameter values $\boldsymbol{\Theta}^*$, written M^* hereafter for brevity. The goal of system identification is to learn a mapping function $\widehat{M}(\mathbf{x}, \mathbf{u}, \widehat{\boldsymbol{\Theta}})$ using a set of training examples $\mathcal{T} = \{(\mathbf{x}(t_k), \mathbf{u}(t_k), y(t_k)), k = 1 \ \ldots \ N\}$. As such, dynamic modeling has always been

challenged by the confluence of two major uncertainties: (i) the form of the model, M, and (ii) the value of the model parameters, $\boldsymbol{\Theta}$. This intertwined confluence of model form and parametric inaccuracy is the Achilles' heel of model development.

Most system identification methods begin by imposing a form for $\widehat{M}$, and, under the assumption that $\widehat{M} \approx M^*$, optimize the parameters $\widehat{\boldsymbol{\Theta}}$ such that the sum of the error, ϵ, between the modeled outputs and the measured observations is minimized, i.e.

$$\epsilon(t_k) = y(t_k) - \hat{y}(t_k, \widehat{M}(\mathbf{x}, \mathbf{u}, \widehat{\boldsymbol{\Theta}})) \tag{1.2}$$

$$\widehat{\boldsymbol{\Theta}} = \arg\min_{\boldsymbol{\Theta}} \sum_{k=1}^{N} L\left(\epsilon(t_k)\right) \tag{1.3}$$

where L is a cost function, e.g. the square function in the case of nonlinear least squares (NLS) [78]. Many of the aformentioned methods of empirical modeling (e.g. neural networks, ARMAX, and support vector machines [59]) are based on this machine learning approach, yet differ in the form of $\widehat{M}$ assumed, the cost function L, and consequently the optimization strategy for estimating Eq. (1.3). In contrast, the methods proposed in the subsequent chapters explicitly search for the form of $\widehat{M}$ that best matches process observations. As we argue in Ch. 2, the mismatch between the true and assumed model structures $\widehat{M} \neq M^*$, is a larger source of error than parametric error for many systems, thus motivating the development of methods that address model structure search.

1.1.1 Genetic Programming in System Identification

GP poses the modeling task more broadly by including the search for the correct model form M^* as part of its optimization strategy, including its parameters. GP attempts to solve the problem

$$\text{minimize } F(M, \mathcal{T}) \quad \text{subject to } M \in \mathfrak{S} \tag{1.4}$$

where $\mathfrak{S}$ is the space of possible models M, and F denotes a minimized fitness function. Given that it is impractical to exhaustively search $\mathfrak{S}$, the model found to minimize $F(M, \mathcal{T})$

may only be locally optimal, but it is assumed that such a model can nevertheless fulfill the purpose of adequately representing the process, as depicted by the measured observations. The task of identifying the model's topology as well as its parameters is known as symbolic regression. In this field, stochastic optimization (or metahueristic) techniques like GP have been successful [126]. These techniques are named for their incorporation of some randomness into the search process, thereby allowing the discontinuous changes needed for exploring equation forms. A number of other techniques fall under this umbrella as well, including hill climbing, simulated annealing, particle swarm optimization [126].

GP is a population-based metaheuristic strategy that uses principles of biological evolution to define the ways in which candidate solutions are updated. Although evolutionary problem-solving strategies were discussed in the sixties [212] and seventies [64], the canonical GP used widely today was developed by John Koza [93] in early nineties (preceded by Cramer [30]), and many of his research practices continue to be popular. The overall process of symbolic regression is visualized in Figure 1.1. In GP, model development is achieved using an evolutionary scheme in which a population of different computer programs are tested for their fitness according to one or several measures of their quality, and go through a process of Darwinian selection and variation to form the next iteration (generation) of candidate solutions. In symbolic regression the programs consist of building blocks (nodes) of equations, for example $\{+,-,*,/,\mathbf{x}\}$, constructed into a graph, most commonly a tree. Candidate solutions in the population are probabilistically selected for survival each generation based on their fitness. Variation is introduced to the search process by mutation and recombination (i.e. crossover) of subprograms in chosen solutions each generation. The search is thus driven by selection and variation: the selection process allows for exploitation of promising solutions while variation mostly serves to explore the neighborhood of promising solutions. This generational process is repeated until an adequate solution is found.

GP and variants of it have been applied successfully for system identification in many fields, including climate modeling [197], robotics [14], mechanics [177], and biological systems [181]. It has become a popular method especially for finding nonlinear differential

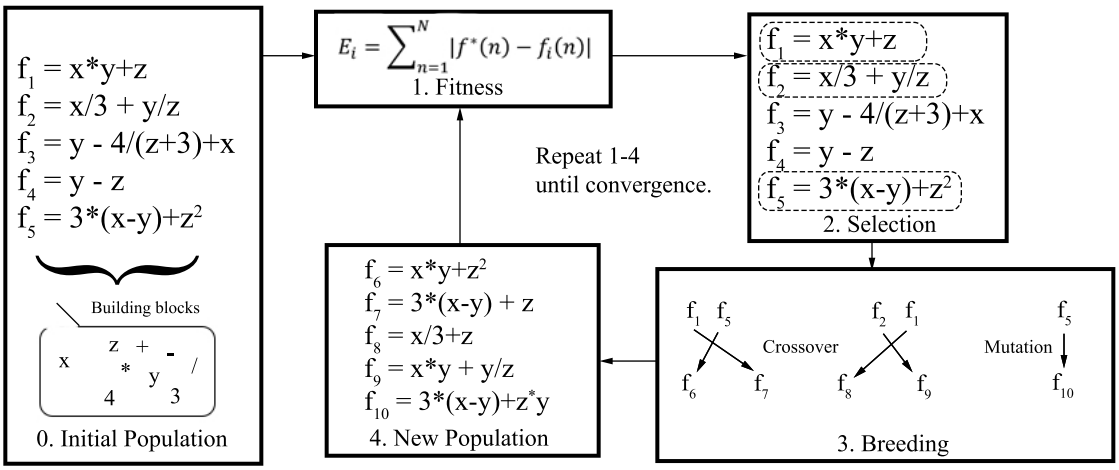

Figure 1.1. Genetic Programming as applied to symbolic regression.

equation solutions [52, 21, 70, 177], in addition to finding closed-form, analytical solutions to them [203, 182]. Furthermore, genetic programming has been used to successfully design systems and solve problems, and has produced human-competitive results in various fields, including antenna design [124], photonics [163], circuit design [94], game play [37], finite algebras [194], and quantum computer programming [192].

Challenges and Extensions

It is important to note that most successful symbolic regression methods extend GP to incorporate other search methods or otherwise deviate from traditional "Koza-style" GP, utlizing techniques such as Pareto-front exploitation [190], coevolution [14, 58], diversity control [128, 178], program size control [69, 158, 187, 15], and concurrent parameter updating [201, 70, 91, 2]. These extensions of standard GP attempt to address some of the common issues impeding its performance, which are described in the following paragraphs.

Traditional GP tends to generate overly complex or over-fit solutions, which are effectively "black box" [197]. A related but not identical problem is bloat, which is the tendency of programs to continue to grow in size with more generations, to the detriment of the search process [187]. Methods to curb program growth include tree snipping [15], parsimony pressure [160], and Pareto-GP with size as an objective [190, 211]. ELGP addresses the problem

of model conciseness by regulating program expression such that accuracy and conciseness are optimized. Unlike the afortmentioned methods, this results in little to no computational overhead compared to traditional GP, as shown in §3.7.

Another issue common to GP is premature convergence, when a solution population converges and stays at a local optimum [159]. This indicates that the entire population has become concentrated in one section of genotype and corresponding phenotype space. Premature convergence is a complex problem because it arises from the intertwined dynamics of selection, which is designed to down-select phenotypes (behaviors) from the population, and variation, which is designed to diversify genotypes (programs) in the population. Thus the mapping of genotypes to phenotypes also plays a central role. As a result, most methods designed to combat premature convergence operate by maintaining diverse genotypes in the population or by maintaining diverse phenotypes in the population. Genotypic diversity can be maintained using deterministic crowding [128], in which individuals compete based on genotypic similarity, or by using structural diversity as an objective [20]. Diversity in the phenotypic sense is the goal of many proposed methods, and is achieved through changes to the selection method or changes to fitness assignment. Examples include lexicase selection [193, 60, 109], implicit fitness sharing [134], age-layering [66, 178], and hierarchical fair competition [68], among others. The proposed ELGP method promotes genotypic diversity by removing selection pressure from silenced portions of programs; a discussion of its effect on behavioral diversity is also included in §3.7.4. Experiments utilizing age-layering and lexicase selection are also used throughout this dissertation.

At least some portion of the aformentioned difficulties with traditional GP may be attributed to the search dynamics of tree representations. Trees are hierarchical structures with strict syntax rules that can easily be violated, e.g. by mismatches of data types or by mutations that result in a mismatch between the number of child nodes a parent node requires to return an argument (i.e. its arity) and the number available in the program. This inflexibility appears at odds with results suggesting that relaxed genotype constraints improve evolvability [170] and expressiveness [196]. Others have suggested that the biolog-

ical features that make evolution powerful and scalable (for example, linear genomes and homologous search operators [195]) are insufficiently emulated by GP [152]. These reasons and others have motivated the development of many different representations, for example stacks [156, 173, 191, 43], linear GP [5, 18], directed acyclic graphs [176], tree adjunct grammars [63], and Cartesian GP [138]. In this work, I make use of stack-based representations for ELGP and M4GP, which makes other salient features of these approaches (epigenetic regulation and multidimensional outputs, respectively) simple to encode.

Although GP has been applied successfully to some binary classification problems (e.g. [210]), until recently [73, 140] it has not been competitive with standard multiclass classification techniques. The exceptions are the recently developed methods M2GP [73] and M3GP [140] that use GP to select and synthesize features and then perform classification in the new feature space using a Mahalanobis distance-based discriminant function. Ch. 4 proposes M4GP, which extends M3GP by using a simple stack representation to encode multidimensional transformations, and implements lexicase selection and age-fitness Pareto survival to improve performance. Whereas M2GP and M3GP were able to tie the performance of some other classification techniques, we show that M4GP surpasses the performance of out-of-the-box classifiers on several problems from different domains.

1.2 Overview of Proposed Methods and their Applications

This dissertation proposes three methods for addressing model structure identification in system identification. The first, MSAM, is motivated by the need to reduce the scope of model search and associated computational expense of GP when a starting model of the system is available. It uses an initial model and function set to optimize the model structure's accuracy with respect to measured data while maintaining the model's intelligibility by constraining updates to the original terms in the model. The second two methods, ELGP and M4GP, build models from scratch, and are tailored to continuous systems ($y \in \mathbb{R}$) and multiclass systems ($y \in \mathcal{C} = \{c_1 \ \ldots \ c_{|C|}\}$), respectively. ELGP proposes a novel representation and learning scheme to provide local model search to GP programs. M4GP re-defines GP programs as

sets of equations that form a wrapper around a distance-based classifier. Although the three methods are quite different, they share as a motivation the focus on model structure search for three different scenarios. In the following section, I will give a brief synopsis of the mechanics of the three methods that are detailed in Chapters 2 to 4, and describe the applications detailed in Chapters 5 to 8.

1.2.1 Model Structure Adaptation

MSAM assumes that a linear-in-parameter starting model of the process has been established, i.e. $\widetilde{M} = \sum_i^Q \tilde{\theta}_i \widetilde{\Psi}_i(\mathbf{x}, \mathbf{u})$. It then adapts this model by optimizing functional couplings applied to the model terms as $\widehat{\Psi}_i \implies \widetilde{\Psi}_i \hat{f}_i^{\gamma_i}$, where $\hat{f}_i$ is chosen from a set of user-defined functions and is modulated by an associated exponent γ_i. As shown in Ch. 2, the exponents associated with the couplings can be optimized in the presence of parametric uncertainty (i.e. when $\widetilde{\mathbf{\Theta}} \neq \mathbf{\Theta}^*$) by quantifying structural changes to the model according to changes in parametric sensitivity. The adaptation utilized by MSAM involves both the choice of $\hat{f}_i$ and the associated exponents, and is guaranteed to only return adaptations that improve the quality of the starting model.

1.2.1.1 Applications

In Ch. 2, MSAM is tested on a set of controlled models that demonstrate its ability to identify correct model structures starting from less complex models. It is then applied to two real world dynamic modeling tasks: the estimation of fluid-structure interaction models and macroeconomic models. In Ch. 5, MSAM is applied to the problem of restructuring proportional, integral, derivative (PID) controllers to allow them to adapt to nonlinear plant dynamics.

1.2.2 Epigenetic Linear Genetic Programming

ELGP proposes the addition of a gene regulation layer (i.e. an epigenetic layer) to linear genomes that can be adapted during the lifetime of individual programs in order to improve the resulting model's accuracy and intelligibility. ELGP has two salient features that improve

its performance: (i) it uses linear, stack-based programs to represent equations, and (ii) it conducts local search of the space of model structures to both improve the fitness and reduce the complexity of models. These feature have been shown to outperform traditional GP on several benchmark regression problems in terms of the conciseness of the developed models, their fitness, and efficiency of the search [110, 107]. The effectiveness of these features is evaluated in Ch. 3 in construction of nonlinear dynamic models.

1.2.2.1 Applications

ELGP is applied to different problems in Chapters 3, 6, 7, and 8. As part of its introduction in Ch. 3, ELGP is tested on hundreds of dynamics problems, including a set of 2nd-order, nonlinear and chaotic systems, a set of randomly constructed ODE systems, and a real word nonlinear dynamics problem regarding the flow of water through cascaded tanks. In Ch. 6, it is used to build models of closed-loop wind turbine dynamics that predict power output and tower loading conditions for an industrial wind turbine. In Ch. 8, the fluid-structure interaction problem introduced in Ch. 2 is revisited using ELGP and a larger set of experiments. A new method of parent selection known as ϵ-lexicase selection [109] is proposed for improving the generalization performance of ELGP for this problem.

1.2.3 Multidimensional Genetic Programming

M4GP is designed to use GP for model structure identification in feature space, as a wrapper to a clustering algorithm. It uses GP to perform a multidimensional transformation of the original attributes and then performs classification using a distance-based discriminant function in the transformed space. In this case, programs consist of a set of transformations $\Phi(\mathbf{x}, \mathbf{u}) = [\phi_1(\mathbf{x}, \mathbf{u}) \ \ldots \ \phi_d(\mathbf{x}, \mathbf{u})]$, and the goal is to find a set of intelligible transformations that produce easily classified clusters in the transformed space. Like ELGP, M4GP utilizes stack-based program encodings and diversity-preserving selection methods to improve its performance relative to previous methods.

1.2.3.1 Applications

M4GP is tested in Ch. 4 on twelve multiclass classification problems that vary in numbers of classes, attributes and data samples. All the problems are real world tests, including a competition data set that is used to recognize human activities from body sensors. Whereas the problems in Ch. 4 are static in nature, M4GP is applied to a real-world dynamics problem in Ch. 7, where it is used to predict the behavior of bald eagles in response to environmental features.

The final chapter of this dissertation (Ch. 9) summarizes the findings of the previous chapters with respect to the methods and their applications. It also takes a look forward at some of the remaining research questions in the hopes of inspiring future work on the subject of model structure identification.

PART I: METHODS

CHAPTER 2

GRADIENT-BASED ADAPTATION OF CONTINUOUS DYNAMIC MODEL STRUCTURES

2.1 Summary

A gradient-based method of symbolic adaptation is introduced for a class of continuous dynamic models[1]. The proposed Model Structure Adaptation Method (MSAM) starts with the first-principles model of the system and adapts its structure after adjusting its individual components in symbolic form. A key contribution of this work is its introduction of the model's parameter sensitivity as the measure of symbolic changes to the model. This measure, which is essential to defining the structural sensitivity of the model, not only accommodates algebraic evaluation of candidate models in lieu of more computationally expensive simulation-based evaluation, but also makes possible the implementation of gradient-based optimization in symbolic adaptation. The proposed method is applied to models of several virtual and real world systems that demonstrate its potential utility.

[1]The work in this chapter was presented at the 2015 ASME Dynamic Systems and Controls Conference [103] and formed the basis of a journal publication [111].

2.2 Introduction

As mentioned in Ch. 1, to improve the intelligibility of adapted models, empirical models in the form of symbolic equations can be formulated by symbolic regression [92, 15, 177, 107]. In symbolic regression, the process variables, inputs, and parameters (constants) are treated as symbols and integrated as blocks to form candidate model structures. Free of restrictions on the form (structure) of candidate model, the search is conducted by genetic programming (GP) for models having best-fit outputs to the measured observations [92]. Even though symbolic regression is computationally expensive, requiring anywhere from thousands to billions of evaluations, it offers a viable approach to modeling poorly understood systems that cannot be readily defined by first-principles models. By the same token, the unrestricted structure of symbolic regression renders it unsuitable for application to better understood systems because of the inherent difficulty of seeding it with starting models [179]. In the absence of a presumed model structure, symbolic regression often yields illegible, albeit accurate, models which do not convey any of the physics of the process.

The Model Structure Adaptation Method (MSAM) proposed in this research contrasts the unrestricted nature of symbolic regression by considering candidate models closely tied to the starting model that are improved by localized gradient-based adaptation. As such, MSAM is designed to remedy the shortcomings of symbolic regression in application to well understood systems for which first-principles models are available. It achieves this by adjusting the individual components of the original model so as to preserve the model's structural integrity, hence, its intelligibility. A key contribution of MSAM, that enables the implementation of gradient-based adaptation, is its use of the model's parameter sensitivity as the measure of 'model difference magnitude'. This measure is used to scale the structural sensitivities such that they will remain robust to parametric error during adaptation.

2.3 Problem Formulation

We consider the model to consist of the weighted sum of individual components Ψ_i, as

$$M = \sum_{i=1}^{Q} \theta_i \Psi_i = \mathbf{\Theta}^T \mathbf{\Psi} \tag{2.1}$$

where $\mathbf{\Psi} = [\Psi_1, \ldots, \Psi_Q]^T$ and each component Ψ_i is the product of any combination of state variables, x_i, included in the state vector $\mathbf{x} = [x_1, \ldots, x_n]^T$, and/or inputs, u_i, in the input vector $\mathbf{u} = [u_1, \ldots, u_m]^T$. For instance, consider the true model of the harmonic oscillator

$$M^* : \ddot{x} = -\frac{c}{m}\dot{x}|\dot{x}| - \frac{k}{m}x^3 + \frac{1}{m}u(t) \tag{2.2}$$

where x denotes its displacement (that is measured), $\dot{x}$ is its velocity, $\ddot{x}$ is its acceleration, $u(t)$ is its input excitation, and m, c, and k denote its mass, damping coefficient, and spring constant, respectively. This true model consists of three components; i.e., $\mathbf{\Psi}^* = [\Psi_1^*,\ \Psi_2^*,\ \Psi_3^*]^T = \left[\dot{x}|\dot{x}|,\ x^3,\ u(t)\right]^T$ with the true parameter values

$$\mathbf{\Theta}^* = \left[\begin{array}{ccc} \theta_1^*, & \theta_2^*, & \theta_3^* \end{array}\right]^T = \left[\begin{array}{ccc} -\frac{c^*}{m^*}, & -\frac{k^*}{m^*}, & \frac{1}{m^*} \end{array}\right]^T$$

Given that the measured outputs have the quality and breadth to characterize the dynamics of the process, the fidelity of the model can be evaluated by how closely the model outputs match the observations [161, 162, 36].

The most common measure of closeness of the model is the magnitude of the prediction error between the process observations, y, and model output, $\hat{y}$, defined as

$$\epsilon(t_k) = y(t_k) - \hat{y}(t_k) = \hat{y}(t_k, \mathbf{\Psi}^*, \mathbf{\Theta}^*, \mathbf{u}) - \hat{y}(t_k, \widehat{\mathbf{\Psi}}, \widetilde{\mathbf{\Theta}}, \mathbf{u}) + \nu \tag{2.3}$$

where $\widehat{\mathbf{\Psi}}$ denotes the candidate model form and $\widetilde{\mathbf{\Theta}}$ the vector of nominal parameter values.

Recall from Ch. 1 that in traditional system identification, model formulation and parameter estimation are performed separately. Once the model form is assumed approximately

correct; i.e., $\widehat{M} \approx M^*$, the model parameters, ascertained identifiable [122], are estimated by minimizing a cost function, V, as

$$\widehat{\boldsymbol{\Theta}} = \arg\min_{\boldsymbol{\Theta}} V = \arg\min_{\boldsymbol{\Theta}} \sum_{k=1}^{N} L\left(\epsilon(t_k)\right) \tag{2.4}$$

where L is a scalar-valued (typically positive) function, such as the square function in non-linear least-squares (NLS). However, when the model form is incorrect (i.e., $\widehat{M} \neq M^*$), parameter estimation either fails or leads to erroneous values associated with an inordinate prediction error, indicating the model mismatch and the need for a better model structure. Therefore, structural accuracy of the model transcends its parametric accuracy, hence the focus of adaptation in MSAM.

The common choice for estimating the model output(s) is numerical integration (i.e., simulation) of state variables. When using different model structures in simulation, the candidate models' output is a function of state variables and inputs, so it varies considerably by even minute changes in the model structure. For instance, because in simulation the rate of output change will depend on the previous outputs, the time span of the transients may change, making it difficult to compare simulated outputs of different models. Therefore, simulation-based estimation of model outputs with different model structures, aside from its computational demand and error propagation tendency, is undesirable for its ambiguity about the model quality.

The alternative to numerical integration is algebraic estimation of candidate model outputs, as commonly performed in symbolic regression [15, 177]. In algebraic evaluation of models, states are estimated from the measured output (by differentiation and/or integration together with various smoothing functions) to yield $\tilde{\mathbf{x}} = [\tilde{x}_1, \ldots, \tilde{x}_n]^T$. The estimated output of the candidate model can then be the state represented by the model; e.g., $\hat{\tilde{x}}$ in the harmonic oscillator of Eq. (2.2). In this configuration, the estimated output finds the form

$$\hat{y}(t_k) = \boldsymbol{\Theta}^T \boldsymbol{\Psi}(\tilde{\mathbf{x}}(t_k), \tilde{\mathbf{u}}(t_k)) \tag{2.5}$$

and the prediction error will have the form

$$\epsilon(t_k) = y(t_k) - \hat{y}(t_k) = \boldsymbol{\Theta}^{*T}\boldsymbol{\Psi}^* - \widetilde{\boldsymbol{\Theta}}^T\widehat{\boldsymbol{\Psi}} \tag{2.6}$$

where $\tilde{\mathbf{u}}$ are the inputs used to produce the measured observations $y(t_k)$. In algebraic evaluation, therefore, model validation is a static test in which the state variables are independent of the model structure and the dependent variable is the modeled variable defined algebraically by the candidate model being evaluated. Referring back to the harmonic oscillator of Eq. (2.2), if the measured variable is the displacement $\tilde{x}(t_k)$, then it can be used to estimate the model velocity $\dot{\tilde{x}}(t_k)$ by numerical differentiation and, say, piece-wise cubic interpolation and loess smoothing [27] to cope with noise and differentiation errors. Now if the candidate model $\widehat{\boldsymbol{\Psi}} = [\dot{x},\ x,\ u(t)]^T$ is the linear form of the harmonic oscillator, and the nominal parameters values are considered to be $\widetilde{\boldsymbol{\Theta}} = \left[-\frac{\tilde{c}}{\tilde{m}},\ -\frac{\tilde{k}}{\tilde{m}},\ \frac{1}{\tilde{m}}\right]^T$, then the model output is estimated as

$$\hat{y}(t_k) = \widehat{\ddot{x}}(t_k) = -\frac{\tilde{c}}{\tilde{m}}\dot{\tilde{x}}(t_k) - \frac{\tilde{k}}{\tilde{m}}\tilde{x}(t_k) + \frac{1}{\tilde{m}}\tilde{u}(t_k)$$

and the prediction error is

$$\epsilon(t_k) = -\frac{c^*}{m^*}\dot{\tilde{x}}(t_k)|\dot{\tilde{x}}(t_k)| + \frac{\tilde{c}}{\tilde{m}}\dot{\tilde{x}}(t_k) - \frac{k^*}{m^*}\tilde{x}(t_k)^3 + \frac{\tilde{k}}{\tilde{m}}\tilde{x}(t_k) + \nu$$

Therefore, the prediction error is not only offset by noise, but also the inaccuracy of the model form; i.e., $\widetilde{\boldsymbol{\Psi}} \neq \boldsymbol{\Psi}^*$ and the parametric error, $\widetilde{\Delta\boldsymbol{\Theta}} = \boldsymbol{\Theta}^* - \widetilde{\boldsymbol{\Theta}}$.

2.4 The Model Structure Adaptation Method

In MSAM, each component is adapted symbolically with the objective of improving the fitness of the model. To this end, a two-stage adaptation strategy is implemented. In the first stage, a comprehensive set of component adjustments is tested after iterative adaptation to select the 'best candidate model'. In the second stage, this best candidate model is adapted further to improve the fitness of the model. The salient features of MSAM are (i) its

unintrusive adjustment scheme which keeps the original model structure intact and conducive to interpretation after adaptation, (ii) its use of gradient-based search for improved efficiency over the stochastic search conducted in symbolic regression, (iii) its capacity to measure the model change magnitude to accommodate gradient-based search in presence of parametric error, and (iv) its ability to find the correct model structure despite parametric inaccuracy.

2.4.1 Adaptation Strategy

In MSAM, the candidate models are formed by adjusting each model component as

$$\widehat{\Psi}_i \implies \widetilde{\Psi}_i \widehat{f}_i(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})^{\gamma_i} \tag{2.7}$$

to yield the candidate model

$$\widehat{M} = \sum_{i=1}^{Q} \widetilde{\theta}_i \widetilde{\Psi}_i \widehat{f}_i(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})^{\gamma_i} = \widetilde{\mathbf{\Theta}}^T \widehat{\mathbf{\Psi}} \tag{2.8}$$

where $\widehat{\mathbf{\Psi}} = \left[\widetilde{\Psi}_1 \widehat{f}_1(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})^{\gamma_1}, \ \ldots, \ \widetilde{\Psi}_Q \widehat{f}_Q(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})^{\gamma_Q}\right]^T$, the $\widehat{f}_i$ are functions of individual state variables or inputs considered to improve the model form, and the γ_i are exponents to achieve two goals: (i) to mitigate the discrete nature of the introduced model change, and (ii) to provide a mechanism for calibrating the degree of change to individual model components for higher granularity. For instance, to achieve $\widetilde{\Psi} = \dot{\tilde{x}} \implies \widetilde{\Psi} f^*(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) = \dot{\tilde{x}}|\dot{\tilde{x}}|$, the adjustment needs to be $\widehat{f}(\mathbf{x}) = |\dot{\tilde{x}}|^{1.0}$. Assuming that the true model can be reached by the introduction of candidate adjustments $\widehat{f}$ to the initial model $\widetilde{\mathbf{\Psi}}$, the true model will have the form $\mathbf{\Psi}^* = \left[\widetilde{\Psi}_1 f_1^*(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})^{\gamma_1^*}, \ \ldots, \ \widetilde{\Psi}_Q f_Q^*(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})^{\gamma_Q^*}\right]^T$. The adaptation strategy, hence, entails applying adjustments of the form (2.7) to individual components of the model $\widetilde{\Psi}$ and then adapting the exponents γ_i to fine-tune the model structure. The goal of MSAM is to first find the form $\widehat{\mathbf{f}} = \left[\widehat{f}_1(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}), \ldots, \widehat{f}_Q(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})\right]^T$, that will match the correct form $\mathbf{f}^* = \left[f_1^*(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}), \ldots, f_Q^*(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})\right]^T$ and then adapt the exponents γ_i, to achieve $\mathbf{\Gamma} = [\gamma_1, \ldots, \gamma_Q]^T \implies \mathbf{\Gamma}^* = \left[\gamma_1^*, \ldots, \gamma_Q^*\right]^T$.

The proposed adaptation strategy, as outlined above, is therefore tailored to starting models with missing couplings. Even though such models may constitute only a subset of

nonlinear models to be envisioned for a process, they encompass considerable nonlinear capacity. Indeed the method could be enhanced in reach by the added provision of introducing new components to the original model. However, that is beyond the premise of the method which is aimed at refining starting models with adequate components to capture the phenomenological aspects of the process. After all, for a more flexible and expanded model structure one could resort to symbolic regression independent of any structural constraints of the starting model. Another requirement of the proposed method is the set of functions f to be introduced into the starting model. While the method does not pose any limitation to the number of functions to be considered, its computational effort will be increased with the larger number of candidate models produced by the expanded set of functions. Given n functions and Q components, the number of possible candidate models would be Q^n. Fortunately, the adaptation of individual candidate models can be performed independently of each other, making parallel execution possible.

Given that finding the correct structural change $\mathbf{f}^*(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})$ transcends adaptation of the corresponding exponents, the first stage of adaptation in MSAM comprises a round robin competition between candidate models of the form (2.8), after a limited number of exponent adaptations, according to a fitness function of the prediction error (Eq. (2.6)). For gradient-based adaptation in the round robin stage, the target output $y(t)$ can be defined by its first-order approximation at the nominal parameter values $\tilde{\theta}_i$, and exponents $\widehat{\gamma}_i$, as

$$y(t) \approx \hat{y}(t, \widehat{\mathbf{\Psi}}, \widehat{\Gamma}, \widetilde{\mathbf{\Theta}}) + \sum_{i=1}^{Q} \widetilde{\Delta\theta}_i \left(\frac{\partial \hat{y}(t, \widehat{\mathbf{\Psi}}, \widehat{\mathbf{\Gamma}}, \widetilde{\mathbf{\Theta}})}{\partial \theta_i} \right) + \sum_{i=1}^{Q} \widehat{\Delta\gamma}_i \left(\frac{\partial \hat{y}(t, \widehat{\mathbf{\Psi}}, \widehat{\mathbf{\Gamma}}, \widetilde{\mathbf{\Theta}})}{\partial \gamma_i} \right) \tag{2.9}$$

where $\widetilde{\Delta\theta}_i = \theta_i^* - \tilde{\theta}_i$ and $\widehat{\Delta\gamma}_i = \gamma_i^* - \hat{\gamma}_i$. The above approximation holds when the structure of the candidate model provides a close first-order approximation of the target output and the partial derivatives of $\hat{y}$ are reasonably close to the corresponding partial derivatives of y.

Ideally, one would want to adapt both the coefficients θ_i and exponents γ_i for each candidate model form $\widehat{\mathbf{\Psi}}$ during the round robin phase with the objective of identifying the correct model form. However, potential collinearity between θ_i/γ_i pairs often hinders their concurrent adaptation, forcing one to adapt the one with the larger influence on the prediction

error. As is discussed in the next section, since the exponents (in the absence of bifurcation) have the larger influence on the prediction error, they are the preferred target for adaptation. Therefore, an underlying assumption of MSAM is that a candidate model with the correct function adjustments $\widehat{\mathbf{f}} = \mathbf{f}^*$ will have the best fitness relative to other candidate models after adaptation despite the parametric error $\widetilde{\Delta\boldsymbol{\Theta}}$.

2.4.2 Precedence of Structural Error to Parametric Error

Albeit anecdotal, the prominence of exponents γ_i over the coefficients θ_i in MSAM is shown via two examples in Fig. 2.1. The two models are those of the harmonic oscillator (left plots) and the van der Pol oscillator (right plots). The plots show the displacements of the two models with different structures and parametric error levels, computed as $\Delta\boldsymbol{\Theta} = \sum_{i=1}^{Q} |\theta_i^* - \tilde{\theta}_i|/\theta_i^*$. Even though three parameters are defined for the van der Pol oscillator, to provide breadth for parameter variation, the parameters of the nominal model ($\ddot{x} = -\eta_1 x^2\dot{x} + \eta_2\dot{x} - \eta_3 x$) are defined as $\eta_1 = \eta_2$ and $\eta_3 = 1$ to match the standard van der Pol oscillator form ($\ddot{x} = \eta\dot{x}(1-x^2)-x$). The results in Fig. 2.1 indicate that both oscillators' displacements are similar in shape with even 100% total parameter error, whereas they differ drastically in shape when their structures change. The exception is the displacement of the oscillator on the right from the model ($\ddot{x} = -\eta_1 x^2\dot{x} + \eta_2\dot{x} - \eta_3 x|x|$) which is very similar to that of the van der Pol oscillator having erroneous parameter values. At these erroneous parameter values, there will be little distinction between the correct and incorrect structures, making it difficult to identify the correct form.

There are three observations to be made of the results in Fig. 2.1. One is that the models' responses are affected more drastically by the structure than the parameter values. This gives credence to prioritizing exponent adaptation over parameter estimation. The second observation is that the shape of the responses are better indicators of structural differences than their magnitudes. The model response shapes are incorporated in the evaluation of models by including the correlation coefficient between the models' responses and their targets in the fitness function. Consideration of output shapes in model evaluation is shown to improve the capacity for structure search in the presence of erroneous parameters and as a precur-

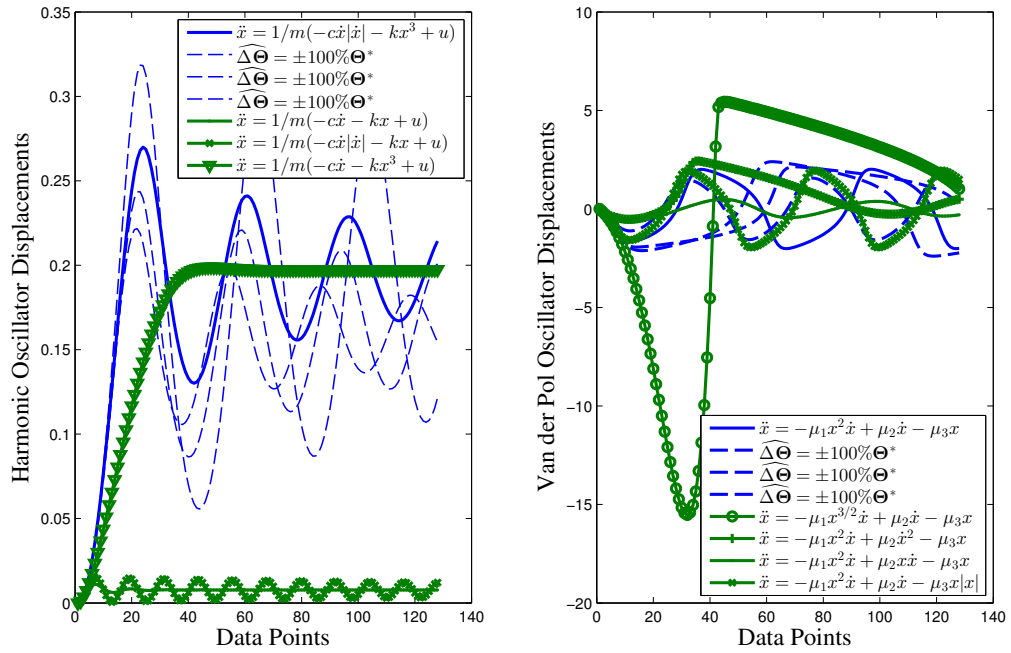

Figure 2.1. Displacements of the harmonic oscillator (left) and the van der Pol oscillator (right) with different parameter values and structures

sor to parameter estimation. The third observation corresponds to the drastic difference of model responses due to the discrete nature of structural differences in Fig. 2.1. Since such drastic differences would be detrimental to gradient-based adaptation, they are mediated by insertion of exponents into the models so as to regulate the size of adaptation.

2.4.3 Scaling of Model Changes

The prominence of structural accuracy to parametric accuracy and the difficulty in concurrent parameter and exponent estimation motivates focusing adaptation on the exponents, rendering the prediction error with the form

$$\epsilon(t) = y(t) - \hat{y}(t, \widehat{\mathbf{\Psi}}, \widetilde{\mathbf{\Theta}}) - \epsilon_\theta \approx \sum_{i=1}^{Q} \widehat{\Delta\gamma}_i \left(\frac{\partial \hat{y}(t, \widehat{\mathbf{\Psi}}, \widetilde{\mathbf{\Theta}})}{\partial \gamma_i} \right) = \epsilon_\gamma = \mathbf{\Phi}_\gamma \widehat{\Delta\mathbf{\Gamma}} \tag{2.10}$$

where ϵ_θ denotes the parametric error approximated by its first-order expansion as

$$\epsilon_\theta \approx \sum_{i=1}^{Q} \widetilde{\Delta\theta}_i \left(\frac{\partial \hat{y}(t, \widehat{\mathbf{\Psi}}, \widetilde{\mathbf{\Theta}})}{\partial \theta_i} \right) \approx \mathbf{\Phi}_\theta \widetilde{\Delta\mathbf{\Theta}} \tag{2.11}$$

and

$$\mathbf{\Phi}_\gamma = \begin{bmatrix} \partial \hat{y}(t_1, \widehat{\mathbf{\Psi}}, \widetilde{\mathbf{\Theta}})/\partial \gamma_1 & \dots & \partial \hat{y}(t_1, \widehat{\mathbf{\Psi}}, \widetilde{\mathbf{\Theta}})/\partial \gamma_Q \\ \vdots & \ddots & \vdots \\ \partial \hat{y}(t_N, \widehat{\mathbf{\Psi}}, \widetilde{\mathbf{\Theta}})/\partial \gamma_1 & \dots & \partial \hat{y}(t_N, \widehat{\mathbf{\Psi}}, \widetilde{\mathbf{\Theta}})/\partial \gamma_Q \end{bmatrix} \tag{2.12}$$

Success of adaptation in MSAM, therefore, relies on the quality of $\mathbf{\Phi}_\gamma$ in Eq. (2.10). Two factors can degrade the estimation of $\mathbf{\Phi}_\gamma$: (i) the presence of parametric error, ϵ_θ, as a bias in the prediction error, and (ii) the non-uniformity of the columns of $\mathbf{\Phi}_\gamma$. As to the first factor, although the parameter error $\widetilde{\Delta\mathbf{\Theta}}$ remains constant during adaptation, the parameter sensitivity matrix $\mathbf{\Phi}_\theta$ varies as a function of $\widehat{\mathbf{\Gamma}}$. This variation causes the bias due to ϵ_θ to be non-constant during adaptation, hence, a shift in the gradient of the objective function. The second factor, namely the non-uniformity of the columns of $\mathbf{\Phi}_\gamma$, stems from the non-uniformity of structural changes $(f_i(\mathbf{x}, \mathbf{u}))^{\gamma_i}$. Unlike typical parameter perturbations that are applied to nonzero parameter values, the values of γ_i initialize at zero to modulate the introduction of functions. As such, their perturbations can have drastic effects on the outputs of the candidate models.

One possible approach to improving the condition of $\mathbf{\Phi}_\gamma$ is to scale the columns of $\mathbf{\Phi}_\gamma$ [6] by the magnitude of model difference caused by the perturbation $\delta\gamma_i$. We quantify the model difference magnitude in terms of parameter sensitivity, according to the following definition.

Definition: The difference between two models of the same structure but a different exponent; i.e., $\mathbf{\Psi}_2 = \widehat{\mathbf{\Psi}}(\mathbf{\Gamma} + \Delta\gamma_i)$ and $\mathbf{\Psi}_1 = \widehat{\mathbf{\Psi}}(\mathbf{\Gamma})$ is quantified by the sum of the ℓ^2-norm of their parameter sensitivity difference over time, as

$$\Delta(\mathbf{\Psi}_2, \mathbf{\Psi}_1) = \sum_{k=1}^{N} \left\| \frac{\partial \hat{y}(t_k, \mathbf{\Psi}_2, \widetilde{\mathbf{\Theta}})}{\partial \mathbf{\Theta}} - \frac{\partial \hat{y}(t_k, \mathbf{\Psi}_1, \widetilde{\mathbf{\Theta}})}{\partial \mathbf{\Theta}} \right\|_2 \tag{2.13}$$

where $\Delta(\boldsymbol{\Psi}_2, \boldsymbol{\Psi}_1)$ is the model difference magnitude, $\partial\hat{y}/\partial\boldsymbol{\Theta} = \left[\begin{array}{cccc} \partial\hat{y}/\partial\theta_1, & \ldots & , \partial\hat{y}/\partial\theta_Q \end{array}\right]$ is the vector of parameter sensitivities at time t_k, and ℓ^2-norm for the vector $\mathbf{v} = [v_1, \ldots, v_n]^T \in \mathbb{R}^n$ is defined as $||\mathbf{v}||_2 = \sqrt{\sum_i^n v_i^2} = (\mathbf{v}^T\mathbf{v})^{\frac{1}{2}}$.

For clarification of the above definition, consider the three harmonic oscillator models of increasing complexity below wherein the acceleration of each model is the estimated output obtained algebraically according to the measured displacement $\tilde{x}(t_k)$.

$$\begin{aligned}
\hat{y}_1(t_k) = \widehat{\ddot{x}}(t_k) &= -\frac{\tilde{c}}{\tilde{m}}\dot{\tilde{x}}(t_k) - \frac{\tilde{k}}{\tilde{m}}\tilde{x}(t_k) + \frac{1}{\tilde{m}}\tilde{u}(t_k); & \boldsymbol{\Psi}_1 &= \left[\dot{\tilde{x}}(t),\ \tilde{x}(t),\ \tilde{u}(t)\right]^T \\
\hat{y}_2(t_k) = \widehat{\ddot{x}}(t_k) &= -\frac{\tilde{c}}{\tilde{m}}\dot{\tilde{x}}(t_k)|\dot{\tilde{x}}(t_k)| - \frac{\tilde{k}}{\tilde{m}}\tilde{x}(t_k) + \frac{1}{\tilde{m}}\tilde{u}(t_k); & \boldsymbol{\Psi}_2 &= \left[\dot{\tilde{x}}(t)|\dot{\tilde{x}}(t)|,\ \tilde{x}(t), \tilde{u}(t)\right]^T \\
\tilde{y}_3(t_k) = \widehat{\ddot{x}}(t_k) &= -\frac{\tilde{c}}{\tilde{m}}\dot{\tilde{x}}(t_k)|\dot{\tilde{x}}(t_k)|^2 - \frac{\tilde{k}}{\tilde{m}}\tilde{x}(t_k) + \frac{1}{\tilde{m}}\tilde{u}(t_k); & \boldsymbol{\Psi}_3 &= \left[\dot{\tilde{x}}(t)|\dot{\tilde{x}}(t)|^2,\ \tilde{x}(t),\ \tilde{u}(t)\right]^T
\end{aligned}$$

Admittedly, $\boldsymbol{\Psi}_3$ is more complex (denoted by $\Uparrow$) than $\boldsymbol{\Psi}_2$, which is more complex than $\boldsymbol{\Psi}_1$; i.e., $\boldsymbol{\Psi}_3 \Uparrow \boldsymbol{\Psi}_2 \Uparrow \boldsymbol{\Psi}_1$. Then according to the above definition, the model difference magnitude $\Delta(\boldsymbol{\Psi}_3, \boldsymbol{\Psi}_1)$ should be as large or larger than $\Delta(\boldsymbol{\Psi}_2, \boldsymbol{\Psi}_1)$ as quantified by the norm of their parameter sensitivity differences; i.e.,

$$\begin{aligned}
\Delta(\boldsymbol{\Psi}_3, \boldsymbol{\Psi}_1) \geq \Delta(\boldsymbol{\Psi}_2, \boldsymbol{\Psi}_1) \implies & \textstyle\sum_{k=1}^N \left|\left| \frac{\partial\hat{y}(t_k, \boldsymbol{\Psi}_3, \widetilde{\boldsymbol{\Theta}})}{\partial\boldsymbol{\Theta}} - \frac{\partial\hat{y}(t_k, \boldsymbol{\Psi}_1, \widetilde{\boldsymbol{\Theta}})}{\partial\boldsymbol{\Theta}} \right|\right|_2 \geq \\
& \textstyle\sum_{k=1}^N \left|\left| \frac{\partial\hat{y}(t_k, \boldsymbol{\Psi}_2, \widetilde{\boldsymbol{\Theta}})}{\partial\boldsymbol{\Theta}} - \frac{\partial\hat{y}(t_k, \boldsymbol{\Psi}_1, \widetilde{\boldsymbol{\Theta}})}{\partial\boldsymbol{\Theta}} \right|\right|_2
\end{aligned}$$

The above inequality can be confirmed analytically, as

$$\begin{aligned}
\left|\left| \frac{\partial\hat{y}(\boldsymbol{\Psi}_3, \widetilde{\boldsymbol{\Theta}})}{\partial\boldsymbol{\Theta}} - \frac{\partial\hat{y}(\boldsymbol{\Psi}_1, \widetilde{\boldsymbol{\Theta}})}{\partial\boldsymbol{\Theta}} \right|\right|_2 &= \sqrt{\left[\dot{\tilde{x}}(t)|\dot{\tilde{x}}(t)|^2 - \dot{\tilde{x}}(t)\right]^2} \geq \\
\left|\left| \frac{\partial\hat{y}(\boldsymbol{\Psi}_2, \widetilde{\boldsymbol{\Theta}})}{\partial\boldsymbol{\Theta}} - \frac{\partial\hat{y}(\boldsymbol{\Psi}_1, \widetilde{\boldsymbol{\Theta}})}{\partial\boldsymbol{\Theta}} \right|\right|_2 &= \sqrt{\left[\dot{\tilde{x}}(t)|\dot{\tilde{x}}(t)| - \dot{\tilde{x}}(t)\right]^2}
\end{aligned}$$

It should be noted here that the symbolic form of parameter sensitivities, shown here for conceptual verification of the above definition, is not necessary for computation of the model difference magnitude, since it can be readily obtained numerically.

As discussed earlier, the model difference magnitude is defined to measure model changes affected by perturbations to the exponents γ_i in Eq. (2.8). To this end, the model difference magnitude is computed for the perturbed model resulted from an exponential perturbation $\delta\gamma_i$ and normalized to render the 'model perturbation magnitude' $\delta\Psi_i$ as

$$\delta\Psi_i = \frac{\sum_{k=1}^{N} \left|\left| \frac{\partial\hat{y}(t_k,\widehat{\mathbf{\Gamma}}+\delta\gamma_i,\widetilde{\mathbf{\Theta}})}{\partial\mathbf{\Theta}} - \frac{\partial\hat{y}(t_k,\widehat{\mathbf{\Gamma}},\widetilde{\mathbf{\Theta}})}{\partial\mathbf{\Theta}} \right|\right|_2}{\sum_{k=1}^{N} \left|\left| \frac{\partial\hat{y}(t_k,\widehat{\mathbf{\Gamma}},\widetilde{\mathbf{\Theta}})}{\partial\mathbf{\Theta}} \right|\right|_2} \tag{2.14}$$

The scaling of structural sensitivity by $\delta\Psi_i$ then takes the form

$$\frac{\partial\hat{y}(t,\widehat{\mathbf{\Gamma}},\widetilde{\mathbf{\Theta}})}{\partial\gamma_i} \approx \frac{\hat{y}(t,\widehat{\mathbf{\Gamma}}+\delta\gamma_i,\widetilde{\mathbf{\Theta}}) - \hat{y}(t,\widehat{\mathbf{\Gamma}},\widetilde{\mathbf{\Theta}})}{\delta\Psi_i} \tag{2.15}$$

wherein the $\delta\Psi_i$ are used in place of $\delta\gamma_i$ in the denominator of the finite difference approximation of the output sensitivity. The availability of the Jacobian $\mathbf{\Phi}_\gamma$ enables estimation of the exponential errors $\Delta\gamma_i$ according to nonlinear least-squares, as

$$\widehat{\Delta\mathbf{\Gamma}} = [\widehat{\Delta\gamma}_1, \ldots, \widehat{\Delta\gamma}_Q]^T = (\mathbf{\Phi}_\gamma^T\mathbf{\Phi}_\gamma)^{-1}\mathbf{\Phi}_\gamma^T\epsilon^N \tag{2.16}$$

and consequent adaptation of the exponents, as

$$\gamma_i(q+1) = \gamma_i(q) + \mu(q)\Delta\gamma_i(q) \tag{2.17}$$

where q is the iteration number and $\mu(q)$ is the adaptation step size, determined at each iteration (see Section 2.5.2).

2.5 Algorithmic Implementation

Given the adjustment strategy in Eq. (2.7), adaptation in MSAM entails finding the adjustments $\mathbf{f}^*$ and their exponents ${\gamma_i}^*$. To this end, adaptation is performed in two stages, wherein improvement of the candidate model(s) is attained through adaptation of the exponents γ_i. In the first stage, using a round robin format, all possible candidate models

are adapted for their viability of improving the model's accuracy with a nominal number of exponent adaptations. Given n adjustments and Q components, the number of candidate models would equal Q^n. The underlying assumption of the round robin stage is that the candidate model with the correct adjustments $\mathbf{f} = \mathbf{f}^*$ will achieve the best fitness in a fixed number of iterations. Therefore, there is always the possibility that MSAM may not find the correct adjustment set because of the limited number of iterations used in the round robin stage. In the second stage, the winner combination is further adapted via its exponents to enhance the model's accuracy.

The adaptation strategy is described in Algorithm 1. Adaptation begins by evaluating the candidate models, composed of unique sets of adjusted components, in a round robin fashion. At the end of the round robin stage, the candidate model associated with the best fitness is chosen for further adaptation in the second stage. For illustration purposes, selection of the best candidate model of the harmonic oscillator in the first stage, followed by its adaptation in the second stage, is shown in Fig. 2.2. The plots in the first stage represent the fitness values of the candidate models during the first 15 iterations of adaptation. The inferior models are discarded for the second stage where adaptation is continued for the best-fit model.

2.5.1 Fitness Function

The fitness function is used mainly to distinguish between the candidate models. Since MSAM emphasizes structural adaptation, it requires a fitness criterion that is more sensitive to structural error than parametric error. A significant indicator of the model structure is the shape of the model output, as was observed in Fig. 2.1. Therefore, included in the fitness function is the correlation coefficient between the model's output and its target so as to represent the closeness of the output's shape to its target [91, 11]. Accordingly, the fitness function in MSAM is defined as

$$F = \frac{\rho(\hat{y}, y)}{\sum_{k=1}^{N} |\epsilon(t_k)|} \tag{2.18}$$

where $\rho(\hat{y}, y)$ denotes the correlation coefficient between the model's output $\hat{y}$ and its target y, computed as $\rho(\hat{y}, y) = C_{\hat{y}y}/\sigma_{\hat{y}}\sigma_y$ where $C_{\hat{y}y}$ is the covariance of $\hat{y}$ and y, and σ denotes

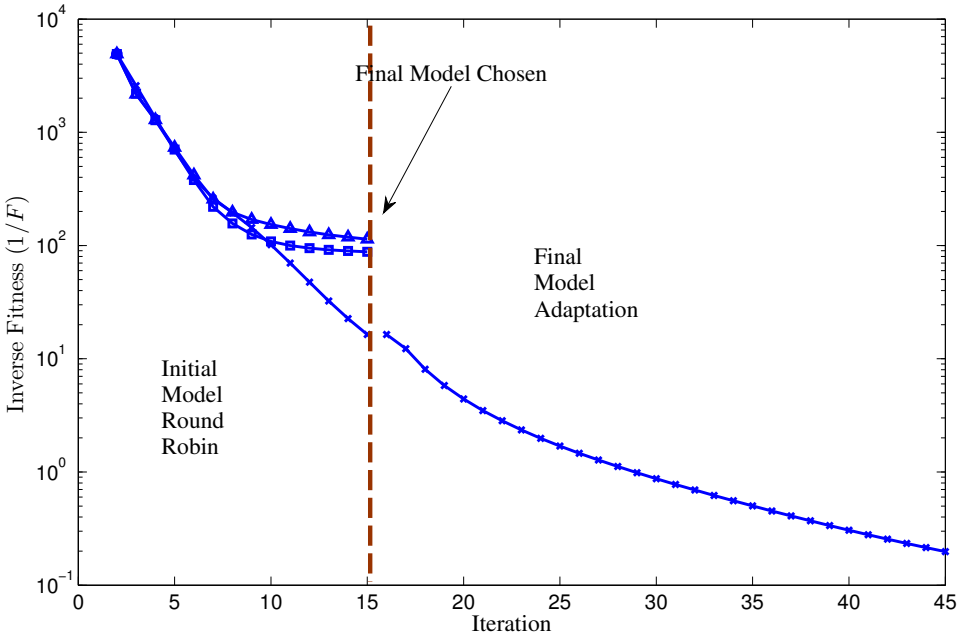

Figure 2.2. Illustration of candidate model selection by MSAM in the round robin stage, followed by further adaptation of the selected model in the second stage, as represented by the inverse of the fitness value for each model

standard deviation. The larger the fitness value the closer the model is to its target, therefore, this fitness function is used primarily to evaluate the fitness of various candidate models in the first stage of adaptation by MSAM.

2.5.2 Selection of Adaptation Step Size

The adaptation step size μ in Eq. (2.17) specifies the confidence in the estimate of $\widehat{\Delta\Gamma}$ from Eq. (2.16). Since this estimate is based on the approximation of the prediction error in Eq. (2.10), its fidelity can be assessed by the accuracy of the prediction error approximation. As a measure of this accuracy, we use the correlation coefficient between the error and its first-order approximation (i.e., the two sides of Eq. (2.10)) to characterize the closeness of approximation of the error shape by the estimated ϵ_γ. Accordingly, the adaptation step size at the iteration q is computed as

$$\mu(q) = \rho\left(\epsilon^N(q), \hat{\epsilon}_\gamma(q)\right) \tag{2.19}$$

where $\epsilon^N = [\epsilon(t_1), \ldots, \epsilon(t_N)]^T$.

2.6 Application Examples

The performance of MSAM is evaluated in two categories: (1) controlled tests, wherein the target model is known and the target data is the simulated output of this model used in lieu of measured observations, and (2) real world tests, wherein the target model is unknown and the measured observations are obtained experimentally. The first tests are intended to examine whether the true underlying model forms can be attained by adaptation. The second tests demonstrate the applicability of the approach to real systems where there are no true models and the preferred models are those that minimize the prediction error.

2.6.1 Controlled Tests

For the controlled validation of the method, three nonlinear models of increasing nonlinearity and order are adapted from simpler starting models. The first model is that of the nonlinear harmonic oscillator, which has been used thus far to illustrate various aspects of the method. It has two model components ($Q = 2$) and two variables ($n = 2$), generating a round robin size of $Q^n = 2^2 = 4$. The second model is that of the van der Pol oscillator, consisting of three components ($Q = 3$) and two variables ($n = 2$). It not only has a larger round robin size than the harmonic oscillator ($3^2 = 9$) but also a nonlinear coupling of velocity and position that needs to be identified. The third model is a third-order state-space model, where each state equation comprises three nonlinear components. As such, this model poses a round robin size of 27 with heavily coupled components in each state equation.

The three models sought by MSAM are shown in Table 2.1 along with the starting models, parameter values, inputs, and perturbation functions. The harmonic oscillator consisted of three components, two of which needed to be adapted to their counterparts in the target model as $\widetilde{\Psi}_1 = \dot{\tilde{x}} \implies \Psi_1^* = \dot{\tilde{x}}|\dot{\tilde{x}}|^{\gamma_1}$ and $\widetilde{\Psi}_2 = \tilde{x} \Longrightarrow \Psi_2^* = \tilde{x}^{\gamma_2}$. For the van der Pol oscillator, the goal was to adapt its first component to $\eta_1 \tilde{x}^2 \dot{\tilde{x}}$, leaving the other two components practically untouched. To guarantee real-valued outputs for the third-order system, the perturbations

were applied as $sign(f_i(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}))|f_i(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})|^{\gamma_i}$. For this model, the three state variables were assumed to be accessible.

Table 2.1. The three models sought by MSAM in the controlled tests

	Harmonic Oscillator	van der Pol Oscillator	3^{rd}-order Model
Target Model	$\ddot{x} = -\frac{c}{m}\dot{x}\lvert\dot{x}\rvert - \frac{k}{m}x^3 + \frac{1}{m}u(t)$	$\ddot{x} + \eta(x^2-1)\dot{x} + x = 0$	$\dot{x}_1 = \theta_1 x_1 x_3 + \theta_2 x_2 x_3 + \theta_3 x_3^2$ $\dot{x}_2 = \theta_4 x_1 x_2 + \theta_5 x_1 x_3 + \theta_6 x_2 x_3$ $\dot{x}_3 = \theta_7 x_1 x_2 + \theta_8 x_1 x_3 + \theta_9 x_2 x_3$
Starting Model	$\ddot{x} = -\frac{c}{m}\dot{x} - \frac{k}{m}x + \frac{1}{m}u(t)$	$\ddot{x} + \eta_1\dot{x} - \eta_2\dot{x} + \eta_3 x = 0$	$\dot{x}_1 = \tilde{\theta}_1 x_1 + \tilde{\theta}_2 x_2 + \tilde{\theta}_3 x_3$ $\dot{x}_2 = \tilde{\theta}_4 x_1 + \tilde{\theta}_5 x_3 + \tilde{\theta}_6 x_2$ $\dot{x}_3 = \tilde{\theta}_7 x_1 + \tilde{\theta}_8 x_3 + \tilde{\theta}_9 x_2$
Parameter Values	$\begin{bmatrix} m^* \\ c^* \\ k^* \end{bmatrix} = \begin{bmatrix} 375 \\ 9800 \\ 130000 \end{bmatrix}$	$\begin{bmatrix} \eta_1^* \\ \eta_2^* \\ \eta_3^* \end{bmatrix} = \begin{bmatrix} 1.5 \\ 1.5 \\ 1 \end{bmatrix}$	$[\theta_1^*, \theta_2^*, \theta_3^*]^T = [\,-3,\ -2,\ -3\,]^T$ $[\theta_4^*, \theta_5^*, \theta_6^*]^T = [\,-3,\ 1,\ -3\,]^T$ $[\theta_7^*, \theta_8^*, \theta_9^*]^T = [\,3,\ 3,\ -1\,]^T$
Excitation	Step input	$\mathbf{x}(0) = [0, -1]$	$\mathbf{x}(0) = [5, 0, 1]$
Perturbance Functions	$\{\ \lvert\dot{\tilde{x}}\rvert,\quad \lvert\tilde{x}\rvert\ \}$	$\{\ \lvert\tilde{x}\rvert,\quad \lvert\dot{\tilde{x}}\rvert\ \}$	$\{\ x_1,\quad x_2,\quad x_3\ \}$

For algorithmic details, let us consider the adaptation of the harmonic oscillator model, in which the options to be considered for adaptation of the first and second components were $\widehat{\Psi}_1 :\ \dot{\tilde{x}} \longrightarrow \dot{\tilde{x}}|\tilde{x}|^{\gamma_1}$ and $\dot{\tilde{x}} \longrightarrow \dot{\tilde{x}}|\dot{\tilde{x}}|^{\gamma_1}$ and $\widehat{\Psi}_2 :\ \tilde{x} \longrightarrow \tilde{x}|\tilde{x}|^{\gamma_2}$ and $\tilde{x} \longrightarrow \tilde{x}|\dot{\tilde{x}}|^{\gamma_2}$. Each of the four models formed from the above options were adapted iteratively by adjusting the exponents γ_1 and γ_2 in 15 iterations. The best model form selected at the end of this first (round robin) stage was further adapted, by improving the exponents γ_1 and γ_2 over 70 more iterations. Adaptation tests were performed with parametric errors ranging from 0% to 200% of the true parameter values. A sample of initial and final outputs before and after structural adaptation is shown in Fig. 2.3. The left plots show the outputs $\widehat{x}$ of the starting model at different levels of parameter error together with the target output. The right plots compare with the target output the outputs of the adapted models at various parametric error levels. The results clearly indicate the effectiveness of MSAM in producing models with outputs that are very close to the target output despite parametric error levels up to 100%.

Even though we use the fitness value as the surrogate, the real goal is the fidelity of the adapted model. To evaluate the reproducibility and accuracy of the model forms achieved by

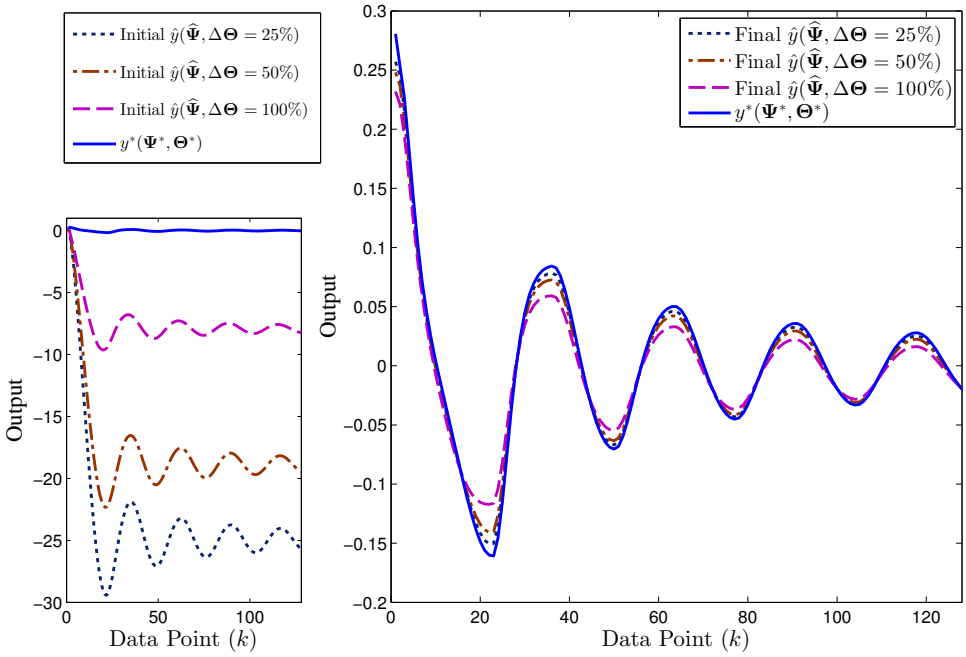

Figure 2.3. Sample of outputs (i.e., $\hat{\hat{x}}$) before (left) and after adaptation (right) by MSAM of the nonlinear harmonic oscillator at three levels of parametric error. The starting model in all cases was $\ddot{x} = \frac{1}{\tilde{m}}(-\tilde{c}\dot{x} - \tilde{k}x + u)$.

MSAM, ten adaptation trials were performed for each of the models with randomly generated parameters at various parametric error levels. The results obtained from these trials at parametric error levels of 0%, 25% and 50% are shown in Table 2.2 for the harmonic and van der Pol oscillators, and in Table 2.3 for the third order model. These tables show the error minimization capacity of MSAM, in terms of the prediction error and correlation coefficient between the estimated and target outputs, and the median final model obtained for each error level. The results indicate that even though the fitness value is influenced by parametric error, the correct model form ($\hat{\mathbf{f}} = \mathbf{f}^*$) is achieved in all cases. The main difference between these models is in exponent values of the final models attained, which deviate from their correct values in accordance with the corresponding parametric error level. Noteworthy in the results is the high level of correlation achieved for the final model outputs at all levels of parameter error ($\rho > 0.999$). The fact that this level of success is not shared by the prediction error validates the lower sensitivity of output shape to parametric error and gives credence to the importance of including the correlation coefficient in the fitness function (see Eq. (2.18)). Noteworthy are the results for the van der Pol oscillator, which indicate that the first component of the final model closely approximates its target ($\eta_1 x^2 \dot{x}$), despite the presence of parametric error. The small exponents associated with the other component adjustments render them negligible. Also noteworthy are near unity values of the correlation coefficients in Table 2.2 between the final model outputs and the target, indicating the lower sensitivity of output shapes to parametric error. Contrary to the correlation coefficient, the prediction error is sensitive to parametric error, as represented by its larger final values at higher parametric error levels. As to the third order model, the third state required a higher number of round robin iterations (40) to robustly select the correct model structure in the presence of parametric error. With the 15 iterations initially used to adapt this state model, MSAM consistently chose the wrong adjustment for its third component. Conversely, the 40 iterations in the round robin stage were adequate to produce consistently correct models at all parametric error levels.

Table 2.2. Performance of MSAM for the nonlinear harmonic oscillator and van der Pol oscillator in terms of the components of the fitness function (i.e., prediction error and correlation coefficient between the estimated and target outputs) and the model forms achieved for different levels of parameter error. The results are from 15 round robin iterations and 70 final choice iterations for the harmonic oscillator and 10 round robin iterations and 75 final choice iterations for the van der Pol oscillator. Ten trials runs were performed at each error level with the parameter values randomly selected. The ± quantities are the standard deviations over the trials.

	Starting		Final		
$\Delta\Theta$	$\sum \lvert\epsilon(t)\rvert$	$\rho(\ddot{\tilde{x}}, \hat{\ddot{x}})$	$\sum \lvert\epsilon(t)\rvert$	$\rho(\ddot{\tilde{x}}, \hat{\ddot{x}})$	Median Final Model
			Harmonic Oscillator		
	Starting Model: $\tilde{m}\ddot{x} + \tilde{c}\dot{x} + \tilde{k}x = u(t)$			Target Model: $m^*\ddot{x} + c^*\dot{x}\lvert\dot{x}\rvert + k^*x^3 = u(t)$	
0% $\boldsymbol{\Theta}^*$	3780	0.7690	0.0534	0.9999	$\tilde{m}\ddot{x} + \tilde{c}\dot{x}\lvert\dot{x}\rvert^{1.00} + \tilde{k}x^{3.00} = u(t)$
25% $\boldsymbol{\Theta}^*$	3685±530	0.769±0.01	0.359±0.25	1.000±0.00	$\tilde{m}\ddot{x} + \tilde{c}\dot{x}\lvert\dot{x}\rvert^{0.984} + \tilde{k}x^{2.991} = u(t)$
50% $\boldsymbol{\Theta}^*$	3603±1049	0.772±0.02	0.689±0.43	0.999±0.00	$\tilde{m}\ddot{x} + \tilde{c}\dot{x}\lvert\dot{x}\rvert^{0.966} + \tilde{k}x^{2.973} = u(t)$
			van der Pol Oscillator		
	Starting Model: $\ddot{x} = -\tilde{\eta}_1\dot{x} + \tilde{\eta}_2\dot{x} - \tilde{\eta}_3 x$			Target Model: $\ddot{x} = -\eta_1^* x^2\dot{x} + \eta_2^*\dot{x} - \eta_3^* x$	
0% $\boldsymbol{\Theta}^*$	276.12	0.446	5.8256	0.9997	$\ddot{x} = -\eta_1\lvert x\rvert^{2.02}\dot{x} + \eta_2\dot{x} - \eta_3 x\lvert\dot{x}\rvert^{0.03}$
25% $\boldsymbol{\Theta}^*$	193.5 ±8.393	0.394 ±0.053	25.484 ±5.524	0.995 ±0.003	$\ddot{x} = -\hat{\eta_1}\lvert x\rvert^{1.934}\dot{x} + \hat{\eta_2}\dot{x}\lvert x\rvert^{0.0226} - \hat{\eta_3} x\lvert\dot{x}\rvert^{0.0634}$
50% $\boldsymbol{\Theta}^*$	205.0 ±21.19	0.250 ±0.0788	37.784 ±13.700	0.9914 ±0.00545	$\ddot{x} = -\hat{\eta_1}\lvert x\rvert^{2.355}\dot{x} + \hat{\eta_2}\dot{x}^{1.132} - \hat{\eta_3} x\lvert\dot{x}\rvert^{-0.046}$

Table 2.3. Performance of MSAM for the three variable SODE in terms of the components of the fitness function (i.e., prediction error and correlation coefficient between the estimated and target outputs) and the model forms achieved for different levels of parameter error. Ten trials runs were performed at each error level with the parameter values randomly selected. The results are from 15 round robin iterations and 100 final choice iterations for states 1 and 2, and 40 round robin iterations and 60 final choice iterations for state 3.

$\Delta\Theta$	Starting		Final		Success	Median Final Model
	$\sum \lvert\epsilon(t)\rvert$	$\rho(\dot{\hat{x}}_i, \hat{\dot{x}}_i)$	$\sum \lvert\epsilon(t)\rvert$	$\rho(\dot{\hat{x}}_i, \hat{\dot{x}}_i)$	Rate	
Starting Model: $\dot{x}_1 = \tilde{\theta}_1 x_1 + \tilde{\theta}_2 x_2 + \tilde{\theta}_3 x_3$						Target Model: $\dot{x}_1 = \theta_1^* x_1 x_3 + \theta_2^* x_2 x_3 + \theta_3^* x_3^2$
0% $\mathbf{\Theta}^*$	3043.217	0.763	403.833	0.993	100%	$\hat{\dot{x}}_1 = \tilde{\theta}_1 x_1 x_3^{1.16} + \tilde{\theta}_2 x_2 x_3^{0.26} + \tilde{\theta}_3 x_3^{2.01}$
25% $\mathbf{\Theta}^*$	3300.194 ±240.851	0.743 ±0.046	440.222 ±120.129	0.998 ±0.002	100%	$\hat{\dot{x}}_1 = \tilde{\theta}_1 x_1 x_3^{1.13} + \tilde{\theta}_2 x_2 x_3^{0.57} + \tilde{\theta}_3 x_3^{2.03}$
50% $\mathbf{\Theta}^*$	3656.362 ±509.102	0.719 ±0.091	747.273 ±239.304	0.995 ±0.017	80%	$\hat{\dot{x}}_1 = \tilde{\theta}_1 x_1 x_3^{1.09} + \tilde{\theta}_2 x_2 x_3^{0.35} + \tilde{\theta}_3 x_3^{2.13}$
Starting Model: $\dot{x}_2 = \tilde{\theta}_4 x_1 + \tilde{\theta}_5 x_3 + \tilde{\theta}_6 x_2$						Target Model: $\dot{x}_2 = \theta_4^* x_1 x_2 + \theta_5^* x_1 x_3 + \theta_6^* x_2 x_3$
0% $\mathbf{\Theta}^*$	5890.721	0.127	49.201	1.000	100%	$\hat{\dot{x}}_2 = \tilde{\theta}_4 x_1 x_2^{1.10} + \tilde{\theta}_5 x_1^{0.99} x_3 + \tilde{\theta}_6 x_2 x_3^{1.10}$
25% $\mathbf{\Theta}^*$	5966.269 ±362.675	0.12672 ±0.019	121.934 ±47.0814	0.998 ±0.0007	100%	$\hat{\dot{x}}_2 = \tilde{\theta}_4 x_1 x_2^{1.11} + \tilde{\theta}_5 x_1 x_3^{0.98} + \tilde{\theta}_6 x_2 x_3^{1.01}$
50% $\mathbf{\Theta}^*$	6062.185 ±730.500	0.124 ±0.038	230.341 ±107.928	0.994 ±0.015	90%	$\hat{\dot{x}}_2 = \tilde{\theta}_4 x_1 x_2^{1.17} + \tilde{\theta}_5 x_1 x_3^{0.87} + \tilde{\theta}_6 x_2 x_3^{0.92}$
Starting Model: $\dot{x}_3 = \tilde{\theta}_7 x_1 + \tilde{\theta}_8 x_3 + \tilde{\theta}_9 x_2$						Target Model: $\dot{x}_3 = \theta_7^* x_1 x_2 + \theta_8^* x_1 x_3 + \theta_9^* x_2 x_3$
0% $\mathbf{\Theta}^*$	4914.037	0.498	88.991	0.999	100%	$\dot{x}_3 = \tilde{\theta}_7 x_1 x_2^{1.10} + \tilde{\theta}_8 x_1^{1.10} x_3 + \tilde{\theta}_9 x_2 x_3^{0.61}$
25% $\mathbf{\Theta}^*$	4795.186 ±375.625	0.508 ±0.033	118.5305 ±29.0678	0.998 ±0.001	100%	$\hat{\dot{x}}_3 = \tilde{\theta}_7 x_1 x_2^{1.12} + \tilde{\theta}_8 x_1^{1.07} x_3 + \tilde{\theta}_9 x_2 x_3^{0.52}$
50% $\mathbf{\Theta}^*$	4683.273 ±730.797	0.518 ±0.063	166.507 ±76.351	0.996 ±0.004	100%	$\hat{\dot{x}}_3 = \tilde{\theta}_7 x_1 x_2^{1.12} + \tilde{\theta}_8 x_1^{1.04} x_3 + \tilde{\theta}_9 x_2 x_3^{0.48}$

Table 2.4. Success rate of MSAM in finding the correct model form for the nonlinear harmonic oscillator at different parametric error levels. Results are reported at the end of the round robin stage from 50 trials of randomly generated parameter values within each parametric error level.

$\Delta\boldsymbol{\Theta}$:	0%	25%	50%	75%	100%	125%	150%	175%	200%
Success Rate:	100%	100%	100%	96%	92%	72%	62%	52%	42%

Crucial to the success of MSAM is selection of the correct model form at the end of the round robin stage. To test this aspect of the performance of MSAM in presence of parametric error, the initial round robin stage was repeated for 50 sets of randomly generated parameters at each level of parametric error up to 200%. Success was declared when the correct model form in terms of adjustments (i.e., $\mathbf{f} = \mathbf{f}^*$) was chosen at the end of the round robin stage. The success rates for different levels of parametric error are shown in Table 2.4. The results indicate that MSAM is completely successful with parametric error levels of up to 50%, more than 90% successful with up to 100% parametric error, and more than 50% successful with up to 175% parametric error. These results underscore the capacity of MSAM in finding the correct model form despite considerable uncertainty in the parameter values, thus obviating concurrent search of both the structure and model parameters.

Even though secondary to proper model form selection, an important aspect of MSAM is adaptation of the exponents of the candidate model during the second stage. The progression of the selected model outputs towards their target for the van der Pol oscillator during second-stage adaptation of the model structure by MSAM is shown in Fig. 2.4 for different levels of parametric error. The plots clearly indicate the convergence of the outputs toward their target despite different levels of parametric error. The only distinction at higher parametric errors is the larger distance between the final and target outputs.

Another important feature of MSAM is the use of $\delta\Psi_i$ from Eq. (2.14) for scaling the columns of $\boldsymbol{\Phi}_\gamma$. A direct ramification of this scaling is the better quality of $\boldsymbol{\Phi}_\gamma$, that will result in better estimates of $\widehat{\Delta\boldsymbol{\Gamma}}$ when used in Eq. (2.16). The improved quality of $\boldsymbol{\Phi}_\gamma$ is illustrated through its condition number ($\lambda_{max}/\lambda_{min}$), computed with and without scaling

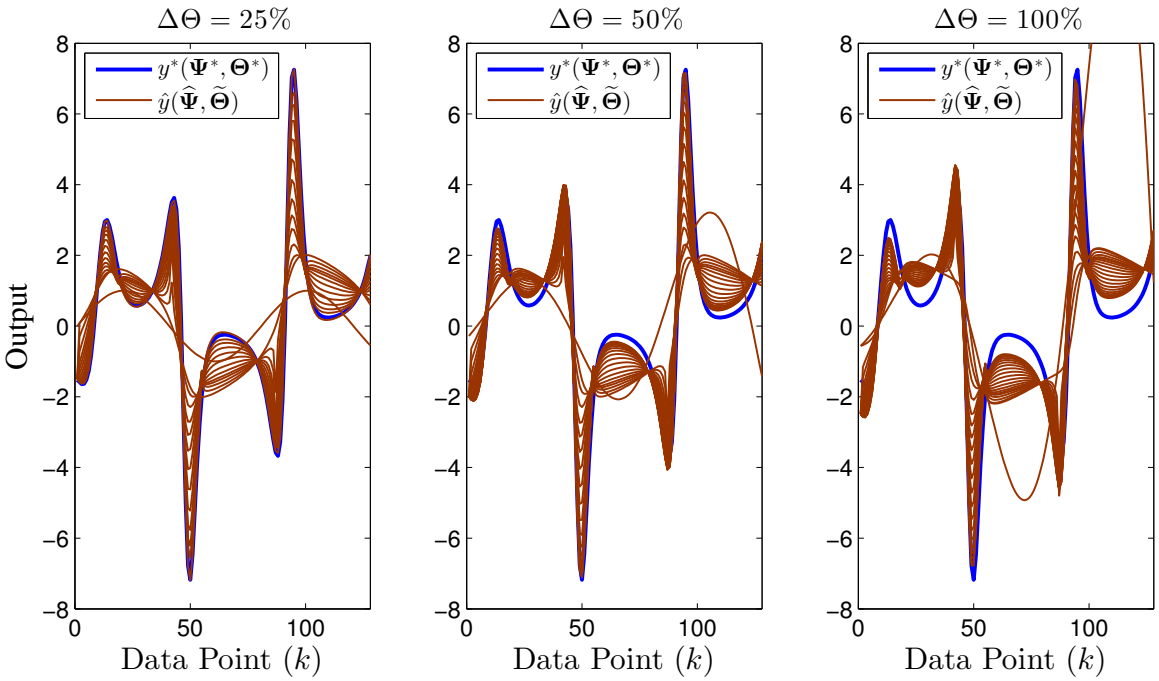

Figure 2.4. Progression of the outputs towards their van der Pol oscillator output target as the model structure is continually adapted by MSAM. The starting model was $\ddot{x} = -\tilde{\eta}_1\dot{x} + \tilde{\eta}_2\dot{x} - \tilde{\eta}_3 x$ with the parameters randomly selected within the parametric error range.

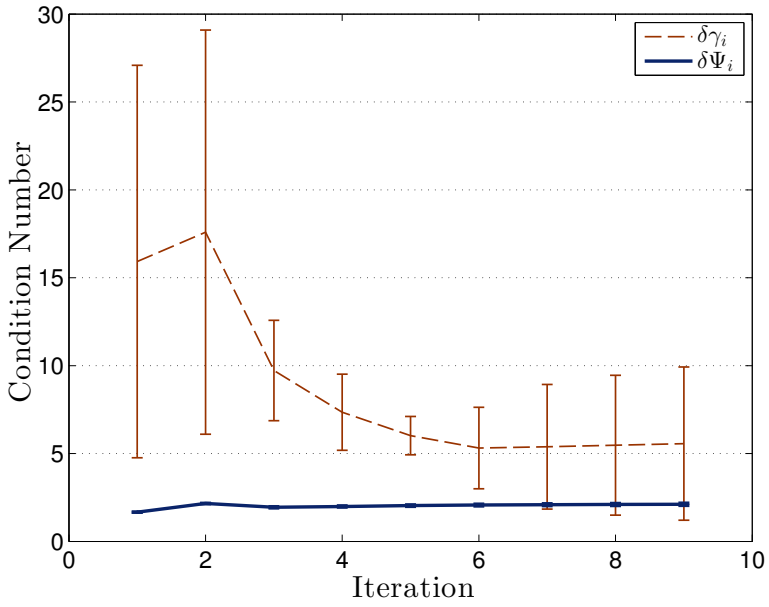

Figure 2.5. Condition number of $\mathbf{\Phi_\Gamma}^T\mathbf{\Phi_\Gamma}$ for various levels of parametric error using the structural sensitivities in Eq. (2.15) with and without scaling by $\delta\Psi_i$. The condition number is calculated for 30 trials of 9 iterations of MSAM at parametric error levels of $\widetilde{\Delta\mathbf{\Theta}} = 25\%$, 50%, and 100%. The error bars represent standard deviation.

by $\delta\Psi_i$ at different levels of randomly generated parametric errors ($\widetilde{\Delta\mathbf{\Theta}}$), as shown in Fig. 2.5. The condition numbers in Fig. 2.5 are much smaller for $\delta\Psi_i$-scaled $\mathbf{\Phi}_\gamma$, and given that the closer the condition number is to unity the more separate (less collinear) are the columns of the matrix [78], the results in Fig. 2.5 clearly indicate the marked improvement in the quality of $\mathbf{\Phi}_\gamma$ scaled by $\delta\Psi_i$. According to the results in Fig. 2.5, not only are the condition numbers nearly always lower for the scaled $\mathbf{\Phi}_\gamma$, but they are also much less sensitive to the parameter error as evidenced by close to zero standard deviations at different levels of parametric error.

2.6.2 Real World Tests

To test MSAM's effectiveness in application to real world cases, target outputs from two sets of experimental data were considered for construction of models. The first set was experimental data obtained for flow-induced vibration of a slender beam. The second set was economic data from the Federal Reserve Economic Data (FRED) (http://research.stlouisfed.org/fred2/).

2.6.2.1 Flow-Induced Vibration

The area of Fluid-Structure Interaction (FSI) offers a pertinent domain for nonlinear modeling of coupled states. Briefly, when a flexible or flexibly-mounted structure is placed in fluid flow, it can move due to the flow forces. The structure's motion changes the flow forces, which in turn affect the structure's motion, constituting an FSI problem. FSI is observed in wind turbines, offshore structures, novel energy extraction ideas and biomedical engineering, among others (e.g., [7, 13, 153, 154, 155, 174, 218]).

For this case study, the experimental data were associated with a uniform cylinder placed in a re-circulating water tunnel, with a test section of 1.27 m $\times$ 0.5 m $\times$ 0.38 m, a turbulence intensity of less than 1% for up to a flow velocity of $U = 0.08$ m/s and a velocity uniformity of less than 2% [185, 184]. The set-up used to hold the cylinder in the test section had two air bearings to reduce the damping and constrain the oscillations of the cylinder to one degree of freedom in the crossflow direction. Springs were attached from the supporting plate holding the cylinder to the fixed housing. The cylinder's displacement and the corresponding flow forces were simultaneously measured at a flow velocity of $U = 0.076$ m/s. The experimental data were split into training and validation sets of 25 seconds in length.

The limit cycle characteristics of the dynamics associated with the displacement ,x, of the cylinder and the applied force, q, motivates the use of the van der Pol oscillator model for representation of q, in lieu of solving the Navier-Stokes equation. Accordingly, the FSI model is considered to be [40]

$$(m_s + 1/4\pi C_M \rho D^2)\ddot{x} + [r_s 2\pi \mathrm{St}(U/D)\rho D^2]\dot{x} + hx = 1/4\rho U^2 D C_{Lo} q \tag{2.20}$$

$$\ddot{q} + \epsilon[2\pi \mathrm{St}(U/D)](q^2 - 1)\dot{q} + [2\pi \mathrm{St}(U/D)]^2 q = (A/D)\ddot{x} \tag{2.21}$$

where St is the Strouhal number, m_s is the mass of the structure, ρ is the fluid density, r_s represents viscous dissipation in the support, γ denotes the stall parameter, U is the free steam velocity, D is the cylinder's diameter, C_M is the added mass coefficient, C_{Lo} is the lift coefficient, and ϵ and A are the van der Pol scaling parameters.

For this study, we focused our analysis on adaptation of the van der Pol oscillator model approximating the vortex force q (Eq. (2.21)). We proceeded first by adapting the model's scaling parameters A and ϵ using non-linear least squares and then used the resulting model as the starting model to be adapted by MSAM. For this adaptation case, 20 round robin iterations were used for selection of the best model, choosing structural perturbations from the set $\left\{ \ |\ddot{q}|, |\dot{q}|, \quad |q| \ \right\}$. The best candidate model was further improved via 20 iterations in the second stage. The resulting model had the form

$$(A/D)\widehat{\ddot{x}} = \tilde{\ddot{q}}|\tilde{\dot{q}}|^{\gamma_1} + \epsilon[2\pi \mathrm{St}(U/D)](\tilde{q}^2|\tilde{\dot{q}}|^{\gamma_2} - |\tilde{q}|^{\gamma_3})\tilde{\dot{q}} + [2\pi \mathrm{St}(U/D)]^2\tilde{q}|\tilde{q}|^{\gamma_4} \tag{2.22}$$

with $\gamma_1 = -0.1178$, $\gamma_2 = -0.0023$, $\gamma_3 = 0.1036$ and $\gamma_4 = 3.2329$.

The results are summarized in Table 2.5 and the initial and final models from MSAM are plotted against the validation data set in Fig. 2.6. The results indicate that whereas parameter estimation of the original model only marginally improves the accuracy of the original model, the structurally adapted model is about 41.7% improved according to magnitude of the prediction error. The VIV modeling problem is considered in more depth in Ch. 8.

Table 2.5. Adaptation of the VIV force equation using experimental results for $U = 0.076$ m/s. The exponents are $\gamma_1 = -0.1178$, $\gamma_2 = -0.0023$, $\gamma_3 = 0.1036$ and $\gamma_4 = 3.2329$.

Model	Training		Validation	
	$\sum \lvert\epsilon(t)\rvert$	$\rho(\ddot{\tilde{x}}, \hat{\tilde{x}})$	$\sum \lvert\epsilon(t)\rvert$	$\rho(\ddot{\tilde{x}}, \hat{\tilde{x}})$
Original Model: $(A/D)\widehat{\ddot{x}} = \ddot{\tilde{q}} + \epsilon[2\pi \mathrm{St}(U/D)](\tilde{q}^2 - 1)\dot{\tilde{q}} + [2\pi \mathrm{St}(U/D)]^2\tilde{q}$	246.53	0.751	243.69	0.769
Parameter-tuned Original Model	245.45	0.780	242.31	0.803
Adapted Model: $(A/D)\widehat{\ddot{x}} = \tilde{\ddot{q}}\lvert\tilde{\dot{q}}\rvert^{\gamma_1} + \epsilon[2\pi \mathrm{St}(U/D)](\tilde{q}^2\lvert\tilde{\dot{q}}\rvert^{\gamma_2} - \lvert\tilde{q}\rvert^{\gamma_3})\tilde{\dot{q}} + [2\pi \mathrm{St}(U/D)]^2\tilde{q}\lvert\tilde{q}\rvert^{\gamma_4}$	136.54	0.890	141.22	0.897
Parameter-tuned Adapted Model	137.20	0.891	141.33	0.898

2.6.2.2 A Macroeconomic Model

The second practical case study was the dynamic model of investment savings (IS) for the United States, which was developed based on economic data from the Federal Reserve Economic Data. The nominal form of the IS model [186] is

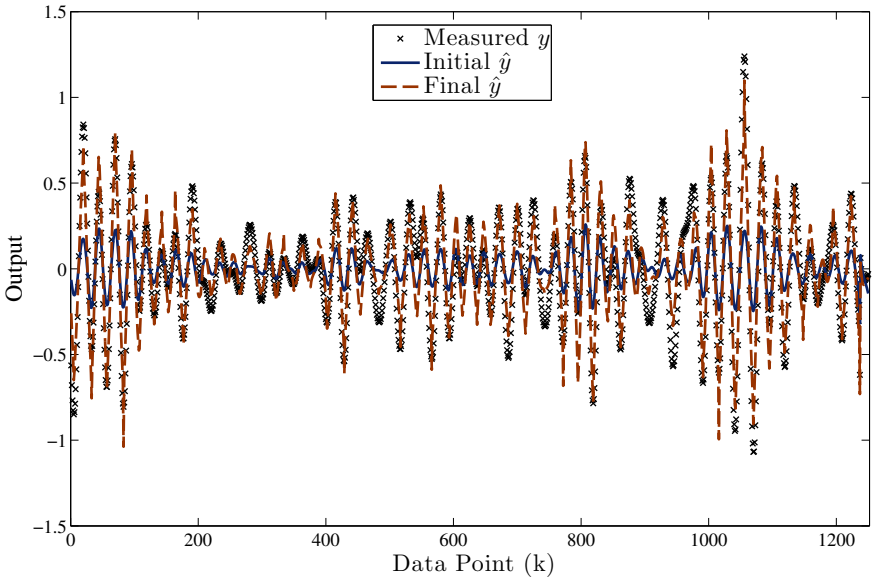

Figure 2.6. Adaptation of the van der Pol form for modeling vortex-induced vibration. The output of the initial and adapted models is shown against the validation data set. The initial $\hat{y}$ is the van der Pol form (Eq. (2.21)) with optimized parameters. The form of the final $\hat{y}$ is given in Table 2.5

$$\dot{y}(t) \quad = \quad \alpha(e(t) - y(t)) \qquad \alpha > 0 \tag{2.23}$$

where e represents total expenditure and y income (gross domestic product (GDP)).

As in the previous case, the above model was parameter-tuned first before being adapted as the starting model by MSAM. The models were trained on historical data from 1959 to 1989, and validated against data from 1990 to 2008. The final model had the form

$$\widehat{\dot{y}}(t) \quad = \quad \hat{\alpha}(\tilde{e}(t)^{\gamma_1} - \tilde{y}(t)|\tilde{y}(t)|^{\gamma_2}) \tag{2.24}$$

where $\tilde{e}$ represented the historical total expenditure data and $\tilde{y}$ the historical income (i.e., GDP). Here, $\gamma_1 = 0.7101$ and $\gamma_2 = -0.4419$.

The fitness values of the model before and after adaptation are shown in Table 2.6, which again indicate that structural adaptation via MSAM leads to better error minimization than parameter estimation on the nominal IS model (Eq. (2.23)). In addition, the adapted model

extrapolates better to unseen data, although some over-fitting can be seen in both models. Parameter tuning of Eq. (2.24) did not yield any improvement in absolute error or correlation, hence its omission from Table 2.6.

Table 2.6. Adapted IS model by MSAM ($\gamma_1 = 0.7101$ and $\gamma_2 = -0.4419$).

Model	Training (1959 - 1989)		Validation (1990 - 2008)	
	$\sum \lvert\epsilon(t)\rvert$	$\rho(\dot{\tilde{y}}, \dot{\hat{y}})$	$\sum \lvert\epsilon(t)\rvert$	$\rho(\dot{\tilde{y}}, \dot{\hat{y}})$
Parameter-tuned Original Model: $\widehat{\tilde{y}}(t) = \hat{\alpha}(\tilde{e}(t) - \tilde{y}(t))$	4944.38	0.519	6361.00	-0.417
Adapted Model: $\widehat{\tilde{y}}(t) = \hat{\alpha}(\tilde{e}(t)^{\gamma_1} - \tilde{y}(t)\lvert\tilde{y}(t)\rvert^{\gamma_2})$	678.40	0.973	2186.98	0.759

2.7 Discussion

The adaptation results presented above demonstrate the effectiveness of MSAM in refining the model topology in presence of parametric uncertainty. The method is found to be effective for parametric errors of up to 50% in the case studies conducted. The error minimization achieved is significant in all cases, even when the best model topology is not chosen due to large parametric errors. Yet the presented results, while they validate MSAM's adaptation strategy, leave several issues to be addressed in future studies. Such issues include the range of MSAM's capacity in coping with parametric error, its computational scalability, its ability to cope with structural collinearity, reachability of potential model forms, consistency of model adaptation across different target measurements, the choice of input excitations, and the significance of measurement noise, as briefly discussed below.

- *Robustness to parametric error:* In the adaptation cases studied so far, MSAM could identify the correct adjustments and adapt the corresponding candidate model to nearly the exact model structure with parametric errors of up to 50%. The robustness of MSAM to parametric uncertainty is contingent upon two factors: (i) that the shape of the candidate models outputs not be substantially affected by parameter error (as illustrated in Fig. 2.1) and (ii) the most accurate parameter values be used for the nominal parameters. To enforce the first factor, the correlation coefficient between

the model output and its target is included in the fitness function to underscore the significance of output shapes in the model selection process. The second factor can be satisfied by performing parameter estimation on the initial model with the hope of improving the parameter values.

- *Scalability:* The scalability of MSAM was demonstrated for systems of up to three adjustments applied to three model components. The main scalability issue is the number of candidate models considered during the round robin stage. Given that with n adjustments applied to Q components Q^n candidate models need to be examined during the round robin phase, the selection process can become overwhelming if the models are examined sequentially. However, candidate models can be evaluated separately and independently of each other. As such, the round robin phase can be run in parallel, reducing the computation time to Q^n/p, with p processors used. Future research could focus on techniques for choosing subsets of round robin models of large-scale problems that cannot be exhaustively searched.

- *Collinearity:* As a gradient-based method, least-squares estimation depends on the non-singularity of the underlying Jacobian. This condition is not satisfied, for instance, seeking the Lotka-Volterra model of inter-species population dynamics, shown as the right hand side model, using the starting model shown in the left

$$\dot{x} = \tilde{a}x - \tilde{b}y - \tilde{c}x \implies \dot{x} = ax - bxy - cx^2$$

$$\dot{y} = \tilde{a}y - \tilde{b}x - \tilde{c}y \implies \dot{y} = ay - bxy - cy^2$$

With this starting model, the first and third columns of the structural sensitivity matrix Φ_γ (Eq. (2.12)) will be collinear for both state equations, due to the sole dependence of the first and third components on x and y, respectively. Given that structural adaptation of both components will be impossible in such a case, a possible recourse would be a sequential approach wherein one component is adapted at a time.

- *Reachability:* In general, MSAM is additive by nature, designed to adapt the potentially inadequate first principles models of the system by the coupling of functions to individual model components. As such, this method is suited to starting models that are less complex than their targets. Furthermore, MSAM is conducted under the assumption that the starting model comprises adequate components for representing the system dynamics, therefore, it provides a sufficient basis for reaching the true model. Accordingly, MSAM's adaptation is restricted to adjustments made to the components of the starting model. One could, indeed, extend the reach of MSAM by expanding the components of the starting model to allow for higher granularity of adjustments, as was demonstrated in the adaptation of the van der Pol oscillator. However, one should be mindful of the fact that such expansions may lead to violating the original premise of MSAM as an efficient yet constrained alternative to symbolic regression.

- *Choice of Input Excitation:* As in all system identification cases, the suitability of the measured output in representing the system is a requisite of search for the true model. As such, the adapted model is only as good as the measurements representing the system. Since in practice one is limited to input excitations that are applicable to the process, the model should be adapted across all measurement sets available from the process to enhance its generality. A focus of our future research is the consistency of adaptation by MSAM when faced with different observations.

- *Measurement Noise:* Measurement noise impedes adaptation by masking the true output of the process (Eq. (1.1)). It also affects algebraic evaluation of candidate models by contaminating the numerical estimates of derivatives of measured observations. Although noise can be addressed to some extent by the application of smoothing and wavelet transforms [133], its presence can be as inhibiting as in other system identification methods.

2.8 Conclusion

A gradient-based method of model structure adaptation is introduced for refinement of starting models of the process. This method, which is designed to increase the nonlinearity of the starting model by adjusting its components, uses exponents of the introduced adjustments to reduce their coarseness as well as to make them conducive to gradient-based search. This method relies on parameter sensitivity of the model to quantify the magnitude of model perturbations for scaling the structural sensitivities. This scaling practice is shown to improve the condition number of the structural sensitivity matrix, thus the search for the correct model structure in presence of parametric uncertainty. Moreover, since the proposed scaling method is independent of the excitation input, it can be used with pre-calculated state variables to preclude simulation-based evaluation of candidate models. Algebraic evaluation of candidate models has the added benefits of allowing state equations to be adapted independently, as in the 3rd order SODE; it is computationally cheaper than numerical integration; and is immune to disruptions of simulation failures. The proposed method is evaluated in controlled tests, wherein the true model is known, as well as in application to real world problems, wherein the measure of success is solely the improved fitness of the adapted model. The results indicate that MSAM finds the correct form of the model in controlled testes despite parametric errors of up to 50% and that it improves the models' fitness by a wide margin in its real world applications.

2.9 Acknowledgments

The authors are grateful to Professor Y. Modarres-Sadeghi for providing the experimental data for the flow-induced vibration example, Professor A. Razmi for pointing to the IS model as a potential example for MSAM and directing us to the associated economic data used for estimating the model. We are also indebted to Professor J. Horowitz and anonymous reviewers for their careful reading of the manuscript and constructive comments.

Algorithm 1 Model Structure Adaptation Algorithm

$Q \longleftarrow$ number of model compartments in $\widehat{M}$
$n \longleftarrow$ number of states (order of the system)
$R \longleftarrow Q^n$
$\hat{M} \longleftarrow$ candidate model
$\xi \longleftarrow$ round robin iteration
$y \longleftarrow$ target output
$\hat{y}_\xi \longleftarrow$ current round robin model output
$\hat{M}_{\xi^*} \longleftarrow$ best round robin model
$\hat{M}_{cb} \longleftarrow$ current best model
$\hat{M}_{best} \longleftarrow$ final best model
$I_1 \longleftarrow$ number of round robin iterations
$I_2 \longleftarrow$ number of winner iterations
$F(\hat{y}_{cb}, y) \longleftarrow F(\hat{y}, y)$
Estimate $\tilde{\mathbf{x}}$
for $\xi \in R$ **do** ▷ Round robin
    $\mathbf{f}_\xi(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) \longleftarrow$ choose set of function perturbations
    $\hat{\Gamma}_\xi = 0$
    $\hat{M}_\xi \longleftarrow \sum_{i=1}^{Q} \tilde{\theta}_i \tilde{M}_i f_i(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})^{\gamma_i}$ ▷ Perturb model compartments
    for $j \in I_1$ **do** ▷ Adapt model $\hat{y}_\xi$
        Calculate $\Phi_\gamma(\hat{y}_\xi)$
        Update $\hat{\Gamma}_\xi$
        Evaluate $\hat{y}_\xi \left(\hat{\Gamma}_\xi\right)$
        if $F(\hat{y}_\xi, y) > F(\hat{y}_{cb}, y)$ **then**
            $\hat{y}_{cb} \longleftarrow \hat{y}_\xi$
            $\mathbf{f}^*(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) \longleftarrow \mathbf{f}_\xi(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})$ ▷ Save best set of function perturbations
        end if
    end for
end for
$\hat{y}_{\xi^*} \longleftarrow \hat{y}_{cb}$
$F(\hat{y}_{cb}, y) \longleftarrow F(\hat{y}, y)$
for $j \in I_2$ **do** ▷ Adapt winning model $\hat{y}_{\xi^*}$
    Calculate $\Phi_\gamma(\hat{y}_{\xi^*})$
    Update $\hat{\Gamma}_{\xi^*}$
    Evaluate $\hat{y}_{\xi^*} \left(\hat{\Gamma}_{\xi^*}\right)$
    if $F(\hat{y}_{\xi^*}, y) > F(\hat{y}_{cb}, y)$ **then**
        $\hat{M}_{cb} \longleftarrow \hat{M}_{\xi^*}$
    end if
end for
$\hat{M}_{best} \longleftarrow \hat{M}_{cb}$

CHAPTER 3

INFERENCE OF COMPACT NONLINEAR DYNAMIC SYSTEMS VIA EPIGENETIC LOCAL SEARCH IN GENETIC PROGRAMMING

3.1 Summary

We introduce a method to enhance the inference of meaningful dynamic models from observational data by genetic programming (GP)[1]. This method incorporates an inheritable epigenetic layer that specifies active and inactive genes for a more effective local search of the model structure space. We define several GP implementations using different features of epigenetics, such as passive structure, phenotypic plasticity, and inheritable gene regulation. To test these implementations, we use hundreds of dynamics problems, including nonlinear ordinary differential equations (ODEs) from several fields of engineering and randomly constructed nonlinear ODE models. The results indicate that epigenetic hill climbing consistently produces more compact dynamic equations with better fitness values, and identifies the exact solution of the system more often, validating the categorical improvement of GP by epigenetic local search. Furthermore, when solving problems with complex dynamics, epigenetic hill climbing reduces the computational effort required to infer the correct underlying dynamics. We then apply the method to the real-world identification of a system of cascaded tanks. We compare the cascaded tanks identification to black-box approaches and analyze the trade-off between accuracy and intelligibility. Finally, we analyze population homology to understand these improvements and show that the epigenetic implementations may provide protection from premature convergence by maintaining diversity in silenced portions of programs.

[1]The work in this chapter was presented at the 2015 GECCO conference [107] and is the basis of a journal publication currently in press [104].

3.2 Introduction

In comparison to system identification methods that presume fixed model structures, symbolic regression can be computationally expensive because of its expanded search space. Furthermore, when guided solely by an error metric, it can yield unwieldy equations from which physical meaning is difficult to extract. To address these potential shortcomings, we present a new method of local search [107] in this chapter that fine-tunes the model structures produced by genetic programming and thereby improves the fitness of models and reduces their complexity. These improvements are achieved by modifying programs in the population at the epigenetic level. Our study of the application of epigenetics to GP is motivated by two main hypotheses: first, that the benefits of epigenetic regulation observed in biology may confer analogous improvements on GP systems; and second, that generalized local search methods can improve the ability of GP to find correct model structures.

In regards to the first hypothesis, even though the expression of biological genes is highly regulated, the role of epigenetics in regulating gene expression is traditionally ignored in GP (with some exceptions, e.g. [43]). However, epigenetic processes may provide several evolutionary benefits. For example, they allow the underlying genotype to encode various expressions and allow neutral variation through crossover and mutation of non-coding segments. This neutrality may allow populations to avoid evolutionary bottlenecks or let them respond to changing evolutionary pressures [77]. In addition, they provide for phenotypic plasticity, which enables gene expression to change in response to environmental pressure [33]. Furthermore, epigenetics allow adaptations to gene expression to be inherited in offspring without explicit changes to the genotype. This legitimizes, via epigenetic processes, once discredited ideas of Lamarck pertaining to the inheritability of lifetime adaptations [77, 65].

Regarding the second hypothesis, local search methods have been developed and integrated into evolutionary algorithms [54, 213, 80, 169, 50], especially in genetic algorithms (GAs), through prescribed changes to the genotype. In GP, especially within the field of symbolic regression, the role of structure optimization is typically left to the GP process while local search is confined to constant optimization via stochastic hill-climbing [15], linear

[71] or non-linear regression [201]. While this improves symbolic regression performance, the methods are inherently limited to problem domains that require constant optimization and cannot be readily applied to other domains (e.g. software synthesis) or aid the search for program topology. Other more generic local search methods, like tree snipping [15], focus on improving secondary metrics like size or legibility. Aside from some recent developments [2], local search is traditionally conducted at the genome level.

The potential evolutionary benefits of epigenetic processes in performing "local search" have motivated the proposed epigenetics-enabled GP system. We propose to conduct topological optimization of programs at the level of gene expression via epigenetic local search. The contributions of this method are twofold: first, it introduces a generic method of topological search of the space of individual genotypes via modifications to gene expression. Second, it improves programs without affecting the genotype and without discarding the acquired knowledge gained through the process, thereby lowering the risk of premature convergence observed in previous studies [213]. It does this by conducting local search on the epigenome rather than the genome and making these adaptations inheritable via evolutionary processes.

The proposed Epigenetic Linear Genetic Programming (ELGP) method is described, tested, and applied to a real-world nonlinear dynamic modeling problem in the following sections. We formulate in §3.3 the identification problem and describe in §3.4 the ELGP method and its application to the inference of dynamic models. We also review the relevant work in the context of GP and nonlinear dynamics modeling in §3.5. We then present the experimental analysis of different epigenetic implementations on a series of increasingly complex problems in §3.6. We begin by testing the method on a large set of simulated nonlinear ODEs from a range of engineering fields, in order to illustrate its breadth of application. To evaluate the scalability of the method in comparison to traditional GP approaches, we then perform identification on hundreds of randomly constructed nonlinear systems, varying in complexity and dimensionality. Finally, we apply ELGP to the real-world identification of the dynamics of cascaded tanks, and analyze its performance in relation to black-box modeling approaches. The results of these experiments are discussed in §3.7. We include as part of

this discussion an analysis of population diversity to study how gene expression evolves for each ELGP implementation.

3.3 Problem Statement

Recall from §1.1 that in the search for the correct model form M^*, GP typically attempts to solve the problem

$$\text{minimize } f(M) \quad \text{subject to } M \in \mathfrak{S} \tag{3.1}$$

where $\mathfrak{S}$ is the space of possible models M, and f denotes a minimized fitness function. Given that it is impractical to exhaustively search $\mathfrak{S}$, the model found to minimize $f(M)$ may only be locally optimal. For practical purposes it is assumed that a sub-optimal model can nevertheless fulfill the purpose of adequately representing the process, as depicted by the measured observations.

A common choice for estimating a candidate model output $\hat{y}(\widehat{M})$ is numerical integration or simulation of the state variables, i.e. the "output error" method [121]. However, given the sensitivity of simulation to different model structures [111] and the computational cost of numerical integration, the alternative approach of algebraically estimating candidate model outputs is preferred for symbolic regression [15, 177]. In the algebraic approach, un-measured states, denoted $\tilde{\mathbf{x}}$, are estimated from measurements via numerical differentiation together with smoothing functions. In the case of first-order differential equations with un-measured state derivatives, the target is estimated numerically as $y(t_k, \mathbf{u}) = \dot{\tilde{x}}$, such that the prediction error of a candidate model has the form

$$\epsilon(t_k) = y(t_k) - \hat{y}(t_k, \widehat{M}(\mathbf{x}, \mathbf{u}, \widehat{\boldsymbol{\Theta}})) = \dot{\tilde{x}}(t_k) - \dot{\hat{x}}(t_k) \tag{3.2}$$

The fitness metric f for individuals is often defined using mean absolute error (MAE) or mean squared error (MSE), although some have argued for using the correlation coefficient due to its insensitivity to linear scaling [83, 91]. We use a fitness metric [111] designed to minimize error and maximize correlation so that both the prediction error and the closeness

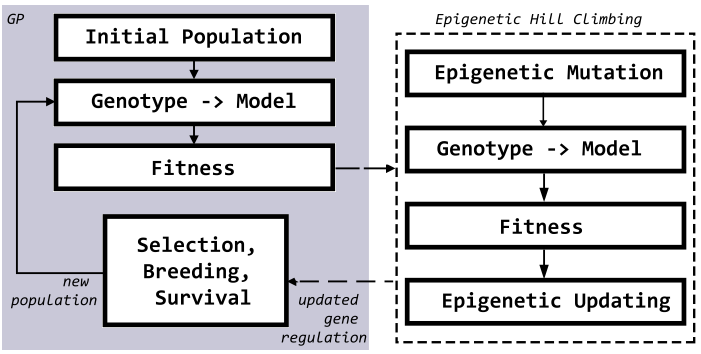

Figure 3.1. Block diagram of ELGP. The typical GP steps are shown on the left. After fitness evaluation and before selection, the population undergoes an iteration of epigenetic hill climbing, represented by the block on the right.

of the output and target shapes, i.e. the coefficient of determination (R^2), can be compared in the results. This fitness metric takes advantage of the covariance comparison afforded by R^2 and avoids the need for post-hoc linear scaling of the solutions, which decreases model conciseness. For target y and output $\hat{y}$, f is defined as:

$$f = \frac{1}{N}\sum_{k=1}^{N} |\epsilon(t_k)|/R^2(y,\hat{y}) \tag{3.3}$$

$$R^2 = \frac{(\mathrm{cov}(y,\hat{y}))^2}{\mathrm{var}(y)\mathrm{var}(\hat{y})} \tag{3.4}$$

3.4 Epigenetic Linear Genetic Programming (ELGP)

In symbolic regression, the search for candidate models is conducted by GP, whereby a population of computer programs that produce models of the process are evolved. Mathematical building blocks compose the genotype of each program that is optimized by an evolutionary algorithm. The operation steps of ELGP[2], outlined in Figure 3.1, start with randomly constructed programs that compose an initial population. The model outputs generated from these programs are evaluated with respect to the training data. Depending on the variant of ELGP as defined in §3.4, the population then undergoes some form of epigenetic adaptation. Afterwards, the population undergoes selection, recombination and mutation, as in standard GP, to produce an updated population, at which point the process repeats until an adequate solution is produced.

[2]source code available from http://www.github.com/lacava

The ELGP method has two salient features that improve its performance: (i) it uses linear, stack-based programs to represent equations, and (ii) it conducts local search of the space of model structures to both improve the fitness and reduce the complexity of models. These feature have been shown to outperform traditional GP on several benchmark regression problems in terms of the conciseness of the developed models, their fitness, and efficiency of the search [110, 107]. The effectiveness of these features is evaluated here in construction of nonlinear dynamic models.

3.4.1 GP Representation

An innovation of the proposed ELGP method is utilization of stack-based representation [156, 196] to accommodate the introduction of epigenetics. In this representation, programs are encoded as post-fix notation, linear genotypes. This stack-based GP system is advantageous because it guarantees syntactic validity for arbitrary sequences of instructions. This property allows instructions to be silenced or activated in a genotype without invalidating the program's ability to execute, in contrast to tree-based representations that can become syntactically invalid due to changes to instructions and literals.

The syntactic robustness of the stack-based approach is achieved mainly by ignoring the execution of instructions that have an arity larger than the current size of the stack. For example, if a + operator attempts to execute and there is only one element on the stack, it does nothing. Furthermore, we base a program's behavior only on the top element of the stack after execution which allows programs to contain unused arguments. These two rules are the key to accommodating diverse program syntax. According to this flexibility, for instance, the genotypes of the following three programs $\mathbf{i_1}$, $\mathbf{i_2}$ and $\mathbf{i_3}$ will produce the identical model $(x_1 + x_2)$:

$$
\begin{aligned}
\mathbf{i_1} &= \begin{bmatrix} x_1 & x_2 & + \end{bmatrix} && \Rightarrow \quad M_1 : (x_1 + x_2) \\
\mathbf{i_2} &= \begin{bmatrix} x_1 & x_2 & + & - & * & / \end{bmatrix} && \Rightarrow \quad M_2 : (x_1 + x_2) \\
\mathbf{i_3} &= \begin{bmatrix} u & + & x_1 & / & x_1 & x_2 & + \end{bmatrix} && \Rightarrow \quad M_3 : (x_1 + x_2)
\end{aligned}
\tag{3.5}
$$

The executions of $-$, $*$ and $/$ in $\mathbf{i_2}$ are ignored due to insufficient stack size, and in $\mathbf{i_3}$, the last element of the executed stack, $(x_1 + x_2)$, is taken as the model. A step-by-step execution of program $\mathbf{i_3}$ is given in Figure 3.2 to illustrate the procedure.

$\mathbf{i_3} = [\ u \quad + \quad x_1 \quad / \quad x_1 \quad x_2 \quad +\ ]$

program execution	stack
1. (u) : push u	$[\ u\ ]$
2. $(+)$: ignore (insufficient arguments)	$[\ u\ ]$
3. (x_1) : push (x_1)	$[\ u \quad x_1\ ]$
4. $(/)$: pull $(x_1), (u)$; push (u/x_1)	$[\ (u/x_1)\ ]$
5. (x_1) : push (x_1)	$[\ (u/x_1) \quad x_1\ ]$
6. (x_2) : push (x_2)	$[\ (u/x_1) \quad x_1 \quad x_2\ ]$
7. $(+)$: pull $(x_2), (x_1)$; push $(x_1 + x_2)$	$[\ (u/x_1) \quad (x_1 + x_2)\ ]$
$\rightarrow$ return last element on stack	$M_3 : (x_1 + x_2)$

Figure 3.2. Stack-based execution of GP program $\mathbf{i_3}$ from Eq. (3.5). Arguments are pushed to the stack and operands ($*$, $+$, etc.) pull arguments from the stack, perform an operation, and push the result. Operands without sufficient arguments are ignored, and the final element on the stack at the end of execution is returned as the model.

3.4.2 Epigenetic Learning and Evolution

We introduce epigenetic information into the GP representation by including an on/off marker on each element in an individual's genotype. This corresponding sequence of on/off markers is referred to as an epigenome. When evaluated together, the expressed program, i.e. model, is produced by executing instructions that are **on** (active) and ignoring the instructions that are **off** (inactive). In this light, one can see that the non-coding genes (known as introns) ignored in programs $\mathbf{i}_2$ and $\mathbf{i}_3$ in Eq. (3.5) provide local solutions to explore in the search space, making it possible to alter the topology and values of the resultant model. For example, program $\mathbf{i_3}$ can admit several models via epigenetic transformations, including

$$
\begin{aligned}
\mathbf{i_3} \rightarrow \mathbf{i'_3} &= \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ u & + & x_1 & / & x_1 & x_2 & + \end{bmatrix} \Rightarrow M'_3 : (u + x_2) \\
\mathbf{i_3} \rightarrow \mathbf{i''_3} &= \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ u & + & x_1 & / & x_1 & x_2 & + \end{bmatrix} \Rightarrow M''_3 : (u/x_1) \\
\mathbf{i_3} \rightarrow \mathbf{i'''_3} &= \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ u & + & x_1 & / & x_1 & x_2 & + \end{bmatrix} \Rightarrow M'''_3 : (u/x_1 + x_2)
\end{aligned}
\tag{3.6}
$$

Similarly, program $\mathbf{i_2}$ in Eq. (3.5) admits the models $(x_1 + x_2)$, $(x_1 - x_2)$, $(x_1 * x_2)$, and (x_1/x_2) via epigenetic transformations.

During the ELGP process depicted in Figure 3.1, the epigenetic markers are initialized randomly (in the initial population) with a probability of being active. We use 50% as the initial probability for the experimental studies in §3.6, chosen according to a previously conducted parametric study [110]. The extent to which epigenetic information is learned and inherited is a research question that we study by exploring different implementations. The simplest topological search method within ELGP is `Ep1M`, which mutates the epigenetic layer of each individual each generation; the hill climber in Figure 3.1 is skipped. Thus for `Ep1M`, epigenetic mutations face only evolutionary pressures. In contrast, the epigenetic hill climbing (EHC) cases `EHC1`, `EHC5` and `EHC10` use the epigenetic information explicitly to improve individuals each generation (the EHC is described in §3.4.2.2). The three methods execute one, five and ten iterations of EHC each generation, respectively. Two control methods, `Base` and `Ep0`, are used for comparison. In the `Base` case, individuals are represented as basic genotypes as in Eq. (3.5). The `Ep0` case acts like `Base` but with half of the genes in the initial code permanently silenced. As such `Ep0` accounts for the effect that passive introns might have. Neither `Base` nor `Ep0` use the right half of the system in Figure 3.1 (i.e. the program never enters epigenetic mutation).

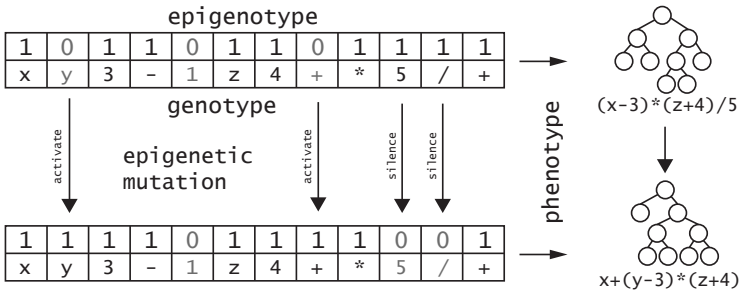

Figure 3.3. Illustration of an epigenetic mutation applied to a GP program. The mutations result in topological changes to the model (phenotype), shown on the right.

3.4.2.1 Epigenetic Mutation

Whitley et al. [213] introduced Lamarckian updating to GAs by conducting local search of the bit strings within 1 Hamming distance of the current bit string. In theory it would be possible to treat the epigenome as a bit string and proceed similarly. However, the cost of GP fitness evaluations may render this approach intractable. Instead, each generation, the epigenome is uniformly mutated with a probability of 10% at each gene. The mutation flips the binary value of the epigenome at the gene, thus activating or silencing that gene. The operation is uniform with respect to the number of instructions. Epigenetic mutation is illustrated in Figure 3.3 to show how these epigenetic changes can result in significant topological changes to the resultant models. For the `EHC1`, `EHC5` and `EHC10` implementations, the epigenetic mutation is followed by hill climbing, described next.

3.4.2.2 Epigenetic Hill Climbing

In order to mimic the acquisition of lifetime learning by epigenetic adaptation, the EHC implementations evaluate epigenetic changes $\mathbf{i} \rightarrow \mathbf{i}'$ to determine whether individuals should be updated. At each iteration of epigenetic mutation, `EHC1`, `EHC5` and `EHC10` test the changes to the model for acceptance. Epigenetic changes to an individual are kept only if the fitness is improved or not changed, i.e. $f_{\mathbf{i}'} \leq f_{\mathbf{i}}$ (fitness f is being minimized).

In addition, we break fitness ties by preferring less complex equations. Model complexity can be represented by several approaches. For example, one can count the number of nodes in

the parse tree, calculate the order of a Chebyshev polynomial fit to the model's output [211], or recursively aggregate the complexity of sub-expressions [90]. Here, we account for model complexity by assigning component function nonlinearities to genotype components [190]. The complexity $C_{\mathbf{i}}$ of program $\mathbf{i}$ with active genotype $\mathbf{g_a} = \begin{bmatrix} g_{a_1} & \dots & g_{a_\ell} \end{bmatrix}$ is defined as $C_{\mathbf{i}} = \sum_{q=1}^{\ell} c(g_{a_q})$, where component function nonlinearities [190] are defined as

$$c(g_a) = \begin{cases} 4 & : \quad (g_a = \log) \vee (g_a = \exp) \\ 3 & : \quad (g_a = \sin) \vee (g_a = \cos) \\ 2 & : \quad (g_a = /) \vee (g_a = \sqrt{}) \\ 1 & : \quad \text{otherwise} \end{cases} \tag{3.7}$$

Lower-complexity programs with equivalent fitness are accepted, giving the condition

$$pass = (f_{\mathbf{i}'} < f_{\mathbf{i}}) \vee ((f_{\mathbf{i}'} = f_{\mathbf{i}}) \wedge (C_{\mathbf{i}'} < C_{\mathbf{i}})) \tag{3.8}$$

If the epigenetically mutated individual $\mathbf{i}'$ does not pass Eq. (3.8), the changes are discarded and $\mathbf{i}$ is kept in the population. Otherwise $\mathbf{i}$ is replaced with $\mathbf{i}'$.

3.4.2.3 Epigenetic Inheritance

A key feature of ELGP is the inheritance of epigenetic values throughout the evolutionary process. During crossover the epigenetic values of the parent genes are kept intact such that the child receives the epigenetic states of the genes it has inherited. If a new gene is introduced via genetic mutation, that gene has the same probability of being active as the initial genes of the population (50%, in the current study).

3.5 Related Work

There has been some work to incorporate epigenetic learning into GP, notably by Tanev [200]. In that case the focus was to model histone modification through a double cell representation as demonstrated in a predator-prey problem. Unlike our approach, Tanev did

not treat lifetime epigenetic modifications as inheritable, as is supported by recent studies in biology [205, 82, 34].

There have also been a number of studies on the effects of non-coding segments in GP, some of which have found that the structural presence of introns protect genotypes from destructive crossover operations (i.e., operations that produce children less fit than their parents) [151, 18]. Non-coding segments were found to be useful in evo-devo for evolution of arbitrary shapes as well [46]. In each of these studies, introns were declared explicitly or measured during evolution, rather than being actively manipulated by the system itself as in ELGP. Our preliminary study of epigenetic initialization finds rates of beneficial crossover to be the highest with the probability set to 50% [110].

In addition, several GP systems use similar stack-based or linear genome representations, such as PushGP [196], Push-forth [84] and Gene Expression Programming [43], that could trivially implement the epigenetic layer incorporated in the ELGP method. Similarly, there are methods that leverage neutrality (i.e., different genotypes with the same fitness) by creating a genotype - phenotype mapping; e.g., Cartesian GP [138] and Binary GP [5]. Our goal with ELGP is to incorporate local search of gene expression as a viable, generic GP extension that does not require large changes to implement. As mentioned earlier, there are a plethora of studies on local search methods for improving GP by Lamarckian or Baldwinian means, yet very few have considered these changes to occur at the epigenetic level instead of the genotype level. A notable exception is Multiple Regression GP [2], in which parameter values are implied at each node location and updated by linear regression. Still, the tangible improvements brought about by this and most other local search methods for symbolic regression are achieved by parametric, rather than topological, search.

Several methods based on GP have been proposed for modeling nonlinear dynamic systems, including ODE model structures [52, 21, 15, 177] and GP-NARMAX models [167, 168]. GP populations are often evolved with multi-objective methods like SPEA2 [220] and NSGA-II [32] to pressure model size. Unlike ELGP, these techniques for compact modeling focus on changes to the core GP algorithm (selection and fitness evaluation). In this regard ELGP

could be readily applied in those proposed frameworks. The multiobjective framework we use in our experiments is that proposed in [178], as discussed in §3.6.1.

More broadly, GP is one approach to nonlinear system identification among many others. A common approach embodied by Hammerstein-Weiner and nonlinear auto-regressive modeling with exogenous inputs (NARX) is to combine a chosen nonlinear transformation (or transformations) with a linear model. Although the use of a nonlinear estimator can increase the capacity compared to ARX modeling, in both cases the structure of the nonlinearity must be specified beforehand, unlike in symbolic regression. These approaches also produce complex models that can obfuscate intuitive explanations of their predictive power. To remedy this, greedy structure selection methods have been proposed for nonlinear polynomial models [56], notably for the NARMAX approach with orthogonal least squares (OLS) [24, 12]. GP methods have also been proposed to optimize the structural identification of OLS models [127]. The goal of ELGP is to improve the ability of GP representations to produce intelligible model structures, which in turn can be applied to auto-regressive representations [105] and coupled with a desired parameter estimation strategy [71, 201, 91].

3.6 Experimental Methods

In this section we describe the evolutionary framework to which ELGP is applied and the settings that are used to conduct the experiments, followed by a description of the set of problems that are used to compare the performance of each GP treatment. §3.6.1 describes the algorithms used to perform selection and search operations within GP, which build upon previous symbolic regression research. In §3.6.2, we describe implementation optimizations related to efficiently performing hill climbing on epigenetically mutated programs. Finally in §3.6.3 we present the set of problems on which ELGP is evaluated, which include simulated ODEs from various fields, randomly constructed ODEs that vary in complexity, and lastly a real-world nonlinear dynamics modeling application to cascaded tanks.

3.6.1 Evolutionary Algorithm

Several state-of-the-art symbolic regression tools leverage Pareto optimization for selection and survival [190, 177], and our preliminary tests (not reported here) confirm that the age-fitness Pareto survival method [178] outperforms traditional GP [93] on several problems. In an effort to demonstrate ELGP on a high-performance configuration, we use age-fitness Pareto survival as the evolutionary algorithm in our experiments. In this scheme, each individual is assigned an age equal to the number of generations since its oldest ancestor was created. Each generation, a new individual is introduced to the population as a means of random restart. Selection for breeding is random, and during breeding a number of children equal to the overall population size is created. At the end of each generation, environmental selection is conducted according to the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [220] to reduce the size of the set P consisting of the current population and the newly created individuals down to the original population size N. Note that the hypervolume method from NSGA-II [32] could also be used for this task, although previous work suggests that SPEA2 performs better in low dimensions [220].

SPEA2 uses two measures to perform this reduction of P: 1) Pareto strength of an individual, $S(\mathbf{i})$, which is the number of individuals equal to or dominated by $\mathbf{i}$, divided by $P + 1$, and 2) a density estimate $D(\mathbf{i}) < 1$, based on the inverse of the distance to the k-th nearest neighbor [189] of $\mathbf{i}$ in objective space (in this case the objectives are normalized between zero and one). These metrics are used to define a fitness value $F(\mathbf{i})$ that combines the total strength of the individuals $\mathbf{j} \in P$ that dominate $\mathbf{i}$, i.e. $\mathbf{j} \prec \mathbf{i}$, with density estimate $D(\mathbf{i})$:

$$F(\mathbf{i}) = \sum_{j \in P, j \prec i} S(\mathbf{j}) + D(\mathbf{i}) \tag{3.9}$$

Every nondominated solution is first copied to the new population. If the new population size is smaller than N, individuals are added in order of lowest $F(\mathbf{i})$. If the population is larger than N, signifying that there are more than N nondominated solutions, individuals are removed iteratively based on $D(\mathbf{i})$. For the latter scenario, the use of $D(\mathbf{i})$ for selection helps preserve spread of solutions along the Pareto front.

Table 3.1. ELGP system settings as applied to the Textbook ODE problems.

General Settings	Value
Population size	1000
Crossover / Mutation	80/20%
Program length limits	[3, 50]
ERC range	[-10,10]
Termination criterion	2.5E10 point evals or $f <$ 1.0E-6
Trials	100
Function set	{ +, −, ∗, /, sin,cos}

Run-time settings for the algorithm are shown in Table 3.1. A uniform alternation crossover operator is used to produce two children from two parents, as in [195]. The mutation operator is applied uniformly to the chosen parent with a probability of 2.5% at each gene. If a constant gene is picked for mutation and ephemeral random constants (ERCs) are being used, the constant is perturbed by Gaussian noise with standard deviation equal to half the magnitude of the constant. Otherwise the instruction is mutated to a randomly chosen gene.

In order to optimize constant values in the models, one iteration of stochastic hill climbing is conducted on model parameters each generation. The hill climber perturbs all constant values in the active genotype by Gaussian noise with a standard deviation equal to 10% of the value of the constant. These changes are kept if they result in a better fitness for the individual. This method of constant optimization is chosen due to its lightweight nature compared to least-squares approaches.

Each trial was allocated a maximum number of point evaluations, i.e., gene executions, to normalize for the different program sizes among methods. A GP run will exit early if the fitness condition $f < 10^{-6}$ is achieved before the designated number of point evaluations has been reached. We observed this fitness termination condition to be sufficient for reaching exact solutions for the problems studied here.

3.6.2 Optimizations

The following optimization provisions are applied to ELGP in order to reduce the number of point evaluations required to evaluate the fitness of an individual that has undergone

epigenetic mutation. The majority of run-time in most GP systems (including ours) is spent in fitness evaluation. This motivates reduction of the number of point evaluations required.

3.6.2.0.1 Fitness Escape EHC requires additional fitness evaluations in order to determine whether the prescribed epigenetic changes will be kept. Given that the fitness $f_{\mathbf{i}}$ of program $\mathbf{i}$ cannot decrease with the evaluation of more fitness cases $k_{1...n}$, evaluation of the epigenetically mutated individual $\mathbf{i}'$ can be halted if at any point $f_{\mathbf{i}'}(1...k_j) > f_{\mathbf{i}}(1...k_n)$ for $1 \leq j < n$. This allows $\mathbf{i}'$ to be discarded before its fitness is fully evaluated because it is guaranteed to be worse than $\mathbf{i}$. Since fitness is always equal to or larger than MAE (see Eq. (3.3)), the halt condition can be defined conservatively using the mean absolute error (MAE) of $\mathbf{i}'$ and the fitness of $\mathbf{i}$ as

$$halt = \frac{1}{N} \sum_{k=1}^{j} |y(t_k) - y(t_k, M_{\mathbf{i}'})| > f_{\mathbf{i}} \tag{3.10}$$

3.6.2.0.2 Stack Tracing In GP tree representations, the output of a node in the program typically depends only on the outputs of its child nodes (and those children's children and so forth). We can say conservatively with ELGP representations that no instruction in the stack is dependent on an instruction to its right. Therefore, when a gene is silenced or activated, only the outputs of the genes to its right in the genotype are affected, hence only part of the program needs to be reevaluated. To avoid repeated instruction evaluations during epigenetic hill climbing, we save the intermediate program outputs of each gene, and after epigenetic mutation reevaluate only those genes to the right of the left-most location of mutation.

Saving the stack outputs is a trade-off between memory and time resources since it requires more memory to save the intermediate outputs but requires fewer point evaluations to evaluate epigenetically mutated individuals. The trade-off is favorable in our implementation because processor resources are much more limited than memory resources. Similar partial evaluation strategies have been proposed, e.g., in [114].

3.6.3 Problems

The methods are first compared on a set of coupled, two-state nonlinear ODEs adapted from [199] and proposed in [180]. Second, they are tested for scalability against a suite of hundreds of randomly generated ODE problems with varying complexity and dimensionality. Finally, ELGP is compared to black-box optimization methods on a real-world problem involving the identification of nonlinear dynamics in a pump-fed system of cascaded water tanks.

3.6.3.1 Textbook ODE problems

The textbook ODE problems represent seven two-state, nonlinear systems from the fields of biology, electrical engineering, physics, ecology, and fluid dynamics. For brevity, the form of the models is shown alongside identification results in Table 3.3. In accordance with [180], each system is simulated for 10 seconds from 4 different initial conditions chosen randomly within stable basins of attraction, giving a total of 400 data points for training. The settings for each problem are summarized in Table 3.1. In order to give a measure of the nonlinearity and/or difficulty of these identification problems, we also use multiple linear regression (`LR`) to estimate models for these systems. The `LR` models are estimated as a weighted sum of the states (in this case the systems have no external inputs), i.e. $\hat{y} = \hat{\beta}^T \mathbf{x}$, where $\hat{\beta}$ is the least-squares solution minimizing $\sum_{k=1}^{N} (y(t_k) - \hat{\beta}^T \mathbf{x}(t_k))^2$.

3.6.3.2 ODE suite

In order to test the scalability of the methods, random ODE systems were generated of varying size (nodes) and dimensionality (number of variables). This approach to scalability testing is used in order to remove problem selection bias and to quantify the methods' performance with different target complexity [176, 28]. The dynamic systems were generated in the following fashion. First, differential equations were randomly generated using the same equation generation technique that initializes populations of equations. Second, the equations were simulated as first-order differential equations (using Runge-Kutta 4) according to a random set of initial conditions chosen from [-5,5]. The output of this simulation was used as the

Table 3.2. ODE suite problem settings.

ODE Suite Settings	
Number of Nodes	3 to 33
Dimensions	1 to 8
Models per setting	5
Trials per model	10
Total models	640
Total trials	6400
Function set	{ +, −, ∗, /, sin,cos,exp,log}

training data set. The validation set was subsequently generated by simulating the equations with initial conditions randomly selected from the range [-10,10]. Equations that produced invalid outputs were discarded. Finally, the valid equations were simplified symbolically in MATLAB in order to determine their most succinct representation, and binned by number of nodes and dimensions. The result of the entire process was 640 unique ODE problems of 3 to 33 nodes and 1 to 8 variables that were subjected to 10 trials of identification, for a total of 6,400 trials per GP treatment. The ODE suite settings are summarized in Table 3.2.

3.6.3.3 Real-world Problem

In order to study the performance of ELGP on a real-world problem, we performed identification based on a set of observations collected from two cascaded tanks fed by a water pump [217]. Using the Bernoulli principle and mass conservation, this system can be represented by the following nonlinear equations:

$$\begin{aligned} \dot{h}_1 &= -\theta_1\sqrt{h_1} + \theta_2 u(t) + w_1(t) \\ \dot{h}_2 &= \theta_1\sqrt{h_1} - \theta_3\sqrt{h_2} + w_2(t) \end{aligned} \tag{3.11}$$

$$\begin{aligned} y_1 &= h_1(t) + e_1(t) \\ y_2 &= h_2(t) + e_2(t) \end{aligned} \tag{3.12}$$

where $\theta_1 = -\frac{a_1\sqrt{2g}}{A_1}$, and $\theta_2 = -\frac{k}{A_1}$, $\theta_3 = \frac{a_2\sqrt{2g}}{A_2}$. States h_1 and h_2 represent the water levels in the upper and lower tanks, respectively; a_1 and a_2 are the outlet areas; A_1 and A_2 are the horizontal cross sections of the tanks; g is the gravitational constant; k is the pump

voltage to flow conversion constant; $w_1(t)$ and $w_2(t)$ are system noise; and $e_1(t)$ and $e_2(t)$ are measurement noise.

The data set comprises 2500 samples, acquired at a sampling period of 5 seconds. We divided this set 50/50 for training and testing. These data have been proposed for benchmarking nonlinear system identification approaches [215] and are freely available [216]. For ELGP, we use `EHC5` with the settings of Table 3.1 and an increased function set $\{+, -, *, /,$ sin, cos, exp, log, $\sqrt{()}\}$.

In order to analyze ELGP's performance in the context of other nonlinear modeling approaches, we compare the accuracy of the ELGP's solution with those of the models developed for this problem. The models developed are an ARX and two NARX with different nonlinear transformations: wavelet networks (NARX-W) and feed-forward neural networks (NARX-NN). We have used the MATLAB System Identification toolbox [123] to generate these models using default settings. Identification of these models was performed as a first-order function of the input, i.e. with the regressors $y_1(t_k - 1)$, $y_2(t_k - 1)$, and $u(t_k)$. For the wavelet network, the set of nonlinear regressors was computed using a radial wavelet expansion with an automatically determined number of terms. For the NARX-NN, a feed-forward network with 10 hidden layers was constructed and trained with back-propagation learning using the Levenberg-Marquardt algorithm. In order to provide uniformity among the predicted outputs of these models and that of ELGP's solution, the ELGP model was simulated on the test set, such that prediction error (Eq. (3.2)) was defined in terms of $\hat{y}_1(t_k)$ and $\hat{y}_2(t_k)$ rather than the state derivatives.

3.7 Results and Discussion

We first present results obtained on the textbook ODE problems by the different methods in §3.7.1. Comparisons include the number of exact solutions, the fitness of the best solutions, and complexity of the best models found by each method. Next in §3.7.2 we analyze the ODE suite results according to fitness as a function of point evaluations in training and testing over the entire suite. To give a sense of the scalability of the methods, we group the results

by target complexity and compare the number of solutions found and the computational effort to reach those solutions. We then compare ELGP to black-box optimization methods on a real world identification problem, the cascaded tanks, in §3.7.3. We analyze the trade-offs between ELGP and other methods in application to this problem, and compare the results with a theoretical model of the system. We end our analysis with a detailed look at population diversity of the Bacterial Respiration problem, which provides insight into the mechanics whereby the variants of ELGP (`Ep1M`, `EHC1`, `EHC5`, `EHC10`), particularly the EHC methods (`EHC1`, `EHC5`, `EHC10`), achieve improved performance on many of these problems.

Table 3.3. The textbook ODE problems (left). The models generated by `Base` and ELGP variant `EHC5` are shown on the right.

System	Target	Base Most Frequent Solution	EHC5 Most Frequent Solution
Bacterial Respiration	$\dot{x} = 20 - x - \frac{x \cdot y}{1+0.5 \cdot x^2}$	$\dot{x} = -(7.006 \cdot (x + 0.4722 \cdot y - 31.11))/y$	$\dot{x} = 20.12 - 1.009 \cdot x - \frac{1.979 \cdot y}{x}$
	$\dot{y} = 10 - \frac{x \cdot y}{1+0.5 \cdot x^2}$	$\dot{y} = 9.994 - 0.5669 \cdot y \cdot sin(3.526/x)$	$\dot{y} = 10.0 - \frac{2.0 \cdot x \cdot y}{2.001+x^2}$
Bar Magnets	$\dot{\theta} = 0.5 \cdot \sin(\theta - \phi) - \sin(\theta)$	$\dot{\theta} = sin(sin(cos(cos(\theta) - \phi) + sin(\theta) - 3.65))$	$\dot{\theta} = 0.5 \cdot sin(\theta - \phi) - sin(\theta)$
	$\dot{\phi} = 0.5 \cdot \sin(\phi - \theta) - \sin(\phi)$	$\dot{\phi} = -0.5 \cdot sin(\theta - \phi) - sin(\phi)$	$\dot{\phi} = -0.5 \cdot sin(\theta - \phi) - sin(\phi)$
Glider	$\dot{v} = -0.05 \cdot v^2 - sin(\theta)$	$\dot{v} = -0.05 \cdot v^2 - sin(\theta)$	$\dot{v} = -0.05 \cdot v^2 - sin(\theta)$
	$\dot{\theta} = v - \cos(\theta)/v$	$\dot{\theta} = v - (cos(\theta))/v$	$\dot{\theta} = v - (cos(\theta))/v$
Lotka-Volterra interspecies dynamics	$\dot{x} = 3 \cdot x - 2 \cdot x \cdot y - x^2$	$\dot{x} = 3.0 \cdot x - 2.0 \cdot x \cdot y - x^2$	$\dot{x} = 3.0 \cdot x - 2.0 \cdot x \cdot y - x^2$
	$\dot{y} = 2 \cdot y - x \cdot y - y^2$	$\dot{y} = 2.0 \cdot y - x \cdot y - y^2$	$\dot{y} = 2.0 \cdot y - x \cdot y - y^2$
Predator Prey	$\dot{x} = x \cdot \left(4 - x - \frac{y}{1+x}\right)$	$\dot{y} = -0.5674 \cdot x \cdot (y + x \cdot cos(\frac{0.7896 \cdot y}{x + cos(0.1632 \cdot x)}) - 6.119)$	$\dot{y} = 0.3516 \cdot x \cdot \left(8.122 - \frac{x}{cos(x/y)} - y\right)$
	$\dot{y} = y \cdot \left(\frac{x}{1+x} - 0.075 \cdot y\right)$	$\dot{y} = (y + 0.01059) \cdot \left(\frac{x}{x+1.004} - 0.07488 \cdot y\right)$	$\dot{y} = y \cdot \left(\frac{x}{1.0+x} - 0.075 \cdot y\right)$
Shear Flow	$\dot{\theta} = \cot(\phi) \cdot cos(\theta)$	$\dot{\theta} = \cot(\phi) \cdot cos(\theta)$	$\dot{\theta} = \cot(\phi) \cdot cos(\theta)$
	$\dot{\phi} = \left(\cos^2(\phi) + 0.1 \cdot \sin^2(\phi)\right) \cdot sin(\theta)$	$\dot{\phi} = 0.45 \cdot sin(\theta) \cdot (cos(2.0 \cdot \phi) + 1.222)$	$\dot{\phi} = 2.076 \cdot sin\left(cos(sin(\phi)) - 0.4918\right) \cdot sin(\theta)$
van der Pol oscillator	$\dot{x} = 10 \cdot \left(y - \frac{1}{3} \cdot (x^3 - x)\right)$	$\dot{x} = (y + 10.41) \cdot (2.232 \cdot sin(x) - 1.879 \cdot x + sin(y))$	$\dot{x} = 10.0 \cdot \left(y - 0.333 \cdot (x^3 - x)\right)$
	$\dot{y} = -\frac{1}{10} \cdot x$	$\dot{y} = -0.1 \cdot x$	$\dot{y} = -0.1 \cdot x$

3.7.1 Textbook ODE problems

As a preliminary evaluation of the intelligibility of ELGP solutions, the most frequent solutions to the textbook ODE problems that were found using `Base` and the ELGP variant `EHC5` are compared against the target model forms in Table 3.3. Of particular note are the differences observed between `Base` and `EHC5` for the Bacterial Respiration (states 1 and 2), Bar Magnets (1), Predator Prey (1 and 2), and van der Pol oscillator (1) identifications. In each of these cases, `EHC5` more often identifies the exact solution, or at least an approximation of it that is less complex than that inferred by `Base`. In some cases, e.g. Bacterial Respiration 1 and van der Pol 1, the approximate solutions from `EHC5` have most of the terms of the target correct, and thus form a sensible approximation of the true system. Note the `Base` solution to Predator Prey 2, i.e.,

$$\dot{y} = (y + 0.01059) \cdot \left(\frac{x}{x + 1.004} - 0.07488 \cdot y \right)$$

could be made more correct through the change $(y + 0.01059) \rightarrow y$. This type of topological model change is easily reached via epigenetic transformations, and, as shown in the Table 3.3, `EHC5` more frequently identifies the underlying target model for this problem.

A central goal of ELGP is also to performance of GP for system identification through local topological search. To this end, the number of solutions, median best fitness (training and test) and average equation size (number of active nodes) for the different methods are summarized in Tables 3.4 and 3.5. Pairwise statistical comparisons are given for each result. Note that three of the identification tasks (Glider 2, Shear Flow 1, and van der Pol 1) are exactly identified 100% of the time by every GP treatment, suggesting that they are easy for GP to solve. For the 11 other problems, the results show that the training and test fitnesses and solution counts are improved by EHC. For example, on each of these 11 problems, the ELGP variant `EHC10` finds significantly ($p < 0.05$) more solutions and produces models with better training and test fitnesses than `Base`, `Ep0` or `Ep1M`, as indicated by highlighting, bold text, and * in the tables. In terms of fitness, `EHC5` provides a significant improvement relative to `Base` on 8 out of 11 and `EHC1` performs better on 5 out of 11 problems, thus suggesting

that results improve with more iterations of EHC. Among the 11 more difficult problems, all of the EHC methods perform significantly better than `Ep0` or `Ep1M`, in terms of fitness as well as exact solutions. Overall the results of `Ep0` and `Ep1M` show a marginal to negative difference in estimation capacity compared to `Base`. The guided search provided by EHC therefore is key to the observed improvements.

The results indicate that every GP method produces better models than `LR` for these problems (with the exception of the linear second state of the van der Pol problem), which is to be expected given the known nonlinear nature of this set of problems. However, `LR` has the advantage of quick training times, with median convergence times on the order of 0.01 second for these problems, as shown in the last column of Tables 3.4 and 3.5. The GP methods converge to the minimum fitness model in approximately 1 to 20 seconds, depending on the difficulty of the problem. It is worth noting that the EHC methods do not increase the computation time compared to `Base`, and occasionally decrease it, which can be attributed to the lower proportion of optimization spent conducting GP generations and the higher proportion spent in EHC. The majority of ELGP computation times are not significantly different from `Base`. These convergence times suggest that the proposed identification methods may be suitable for certain online applications, depending on the time window constraints between model updates.

In addition to improving predictive ability, a motivation for the ELGP design is the delivery of concise solutions. This property is evident from the average program sizes in Tables 3.4 and 3.5. To further evaluate this aspect of the results, in Figure 3.4, the best solutions of the 100 trials are evaluated in terms of *Solution Bloat*, defined as the difference in complexity (Eq. (3.7)) between the GP solution and the target. These results show that the ELGP variants all produce solutions that are more succinct than those achieved by `Base`. Among ELGP variants, `Ep0` and `Ep1M` produce the most succinct models. The EHC methods also produce more succinct models with less solution bloat than `Base`; however, we observe that the hill climbing aspect of EHC leads to slightly larger models than `Ep0` or `Ep1M`.

Nevertheless, in addition to producing succinct models than `Base`, the EHC methods find exact solutions more often for most problems, as shown in Tables 3.4 and 3.5.

3.7.2 ODE suite

To evaluate the ability of ELGP to scale to problems of increasing complexity, we evaluate the performance of the treatments `Base`, `Ep0`, `Ep1M`, `EHC1` and `EHC5` on a suite of 640 randomly generated target systems. The best fitness on training and test sets for the entire ODE suite is shown in Figures 3.5 - 3.6 as a function of point evaluations. The results indicate the better fitness minimization properties of the ELGP variants `EHC1` and `EHC5`. The number of solutions found, and the computational effort to reach those solutions, are plotted in Figures 3.7 and 3.8, respectively. The results are broken into groups based on the number of arguments and operands (i.e. nodes) in the solution equation, for example 7, 9, 11, and so on. Figure 3.7 demonstrates the ability of the ELGP variants, especially `EHC5`, to find more solutions than `Base` or `Ep0`, and the difference in performance grows as the number of nodes in the solution increases, although the overall number of solutions decreases with more complex targets. In addition, `EHC1` and `EHC5` tend to find solutions with less computational effort than the other methods, as demonstrated in Figure 3.8, where we plot the number of point evaluations to termination for the trials in which exact solutions were found. The results also suggest that the computational effort improvement afforded by EHC increases with the complexity of the problem.

3.7.3 Real-world Problem

In this section, `EHC5` and several standard modeling approaches are applied to the identification of a benchmark system of cascaded tanks fed by a pump [216]. Since the true parameters of the system are not provided in [216], we first estimate the parameters of the model in Eq. (3.12) by linear regression, which yields

$$\begin{aligned} \dot{\hat{y}}_1 &= -0.0122\sqrt{\hat{y}_1} + 0.0188\ u(t) \\ \dot{\hat{y}}_2 &= -0.0481\sqrt{\hat{y}_1} + 0.0452\sqrt{\hat{y}_2} \end{aligned} \tag{3.13}$$

Table 3.4. Comparison of best-of-run results for the ODE benchmark problems. Statistical significance ($p < 0.05$) is denoted as follows: $^{\diamond}$: better than `MR`; (·): better than `Base`; **bold**: better than `Ep0`; $*$: better than `Ep1M`; †: better than `EHC1`; ‡: better than `EHC5`. Exact solution p-values are based on pairwise chi-squared tests with Holm correction. Fitness and bloat p-values are based on pairwise Wilcoxon rank-sum tests.

Problem	Method	Exact Solutions (%)	Median Best Fitness	Median Program Size	Median Convergence Time (s)
Bacterial Respiration 1	MR	0	0.21196	n/a	†‡*0.01
	Base	0	$^{\diamond}$**0.14519**	15.61	38.77
	Ep0	0	$^{\diamond}$0.16068	†‡ 10.33	35.35
	Ep1M	0	$^{\diamond}$0.15657	†‡ 10.42	50.42
	EHC1	0	*$^{\diamond}$**0.14160**	‡ 11.91	46.14
	EHC5	0	*$^{\diamond}$**0.13907**	12.52	29.03
	EHC10	0	†*$^{\diamond}$**0.13245**	13.00	52.35
Bacterial Respiration 2	MR	0	0.21273	n/a	†‡*0.01
	Base	0	$^{\diamond}$0.00785	15.54	74.09
	Ep0	0	$^{\diamond}$0.00794	†‡ 10.29	31.31
	Ep1M	0	$^{\diamond}$0.00957	†‡ 10.29	71.70
	EHC1	0	*$^{\diamond}$**0.00741**	‡ 11.58	94.61
	EHC5	0	†*$^{\diamond}$**0.00634**	12.19	77.55
	EHC10	0	†*$^{\diamond}$**0.00617**	12.19	75.22
Bar Magnet 1	MR	0	0.04057	n/a	†‡*0.01
	Base	2	†*$^{\diamond}$**0.02351**	14.84	87.13
	Ep0	1	$^{\diamond}$0.02866	†‡ 10.32	74.07
	Ep1M	2	$^{\diamond}$0.03155	†‡ 10.35	83.61
	EHC1	7	*$^{\diamond}$**0.02733**	‡ 11.31	48.95
	EHC5	7	†*$^{\diamond}$**0.02242**	12.24	***39.49**
	EHC10	7	†*$^{\diamond}$**0.02103**	12.64	***50.79**
Bar Magnet 2	MR	0	1.19817	n/a	‡0.01
	Base	$^{\diamond}$18	*$^{\diamond}$**0.01254**	15.51	68.79
	Ep0	$^{\diamond}$16	$^{\diamond}$0.01502	†‡ 10.77	61.54
	Ep1M	$^{\diamond}$16	$^{\diamond}$0.01729	†‡ 10.72	81.35
	EHC1	*$^{\diamond}$**41**	*$^{\diamond}$**0.00005**	11.59	68.86
	EHC5	*$^{\diamond}$**45**	*$^{\diamond}$**0.00000**	12.00	93.17
	EHC10	*$^{\diamond}$**38**	*$^{\diamond}$**0.00001**	12.34	‡80.07
Glider 1	MR	0	4.52904	n/a	0.01
	Base	$^{\diamond}$75	$^{\diamond}$0.00000	7.22	82.65
	Ep0	$^{\diamond}$77	$^{\diamond}$0.00000	5.07	91.69
	Ep1M	$^{\diamond}$76	$^{\diamond}$0.00000	4.25	64.20
	EHC1	*$^{\diamond}$**99**	*$^{\diamond}$**0.00000**	* **3.03**	53.92
	EHC5	*$^{\diamond}$**100**	*$^{\diamond}$**0.00000**	* **3.26**	73.33
	EHC10	*$^{\diamond}$**100**	*$^{\diamond}$**0.00000**	**3.53**	81.55
Glider 2	MR	0	0.53583	n/a	0.01
	Base	100	$^{\diamond}$0.00000	2.76	19.89
	Ep0	100	$^{\diamond}$0.00000	‡ 1.71	7.61
	Ep1M	100	$^{\diamond}$0.00000	‡ 1.62	6.11
	EHC1	100	$^{\diamond}$0.00000	‡ 1.49	4.99
	EHC5	100	$^{\diamond}$0.00000	2.46	6.37
	EHC10	100	$^{\diamond}$0.00000	‡ 1.57	4.14
Lotka-Volterra 1	MR	0	0.94071	n/a	††0.01
	Base	7	*$^{\diamond}$**0.04218**	16.30	48.83
	Ep0	$^{\diamond}$10	$^{\diamond}$0.06897	†‡ 11.07	83.16
	Ep1M	8	$^{\diamond}$0.06805	†‡ 11.24	59.40
	EHC1	$^{\diamond}$13	*$^{\diamond}$**0.00050**	‡ 12.71	82.81
	EHC5	*$^{\diamond}$**30**	†*$^{\diamond}$**0.00003**	13.20	73.86
	EHC10	†*$^{\diamond}$**43**	††‡*$^{\diamond}$**0.00000**	13.18	67.98

Table 3.5. Comparison of best-of-run results for the ODE benchmark problems. Statistical significance ($p < 0.05$) is denoted as follows: $^{\diamond}$: better than `MR`; $(\cdot)$: better than `Base`; **bold**: better than `Ep0`; $*$: better than `Ep1M`; †: better than `EHC1`; ‡: better than `EHC5`. Exact solution p-values are based on pairwise chi-squared tests with Holm correction. Fitness and bloat p-values are based on pairwise Wilcoxon rank-sum tests.

Problem	Method	Exact Solutions (%)	Median Best Fitness	Median Program Size	Median Convergence Time (s)
Lotka-Volterra 2	MR	0	0.19556	n/a	0.01
	Base	$^{\diamond}$53	$^{\diamond}$0.00000	14.51	89.27
	Ep0	$^{\diamond}$73	$^{\diamond}$**0.00000**	**6.33**	59.00
	Ep1M	$^{\diamond}$67	$^{\diamond}$0.00000	7.04	77.29
	EHC1	$*^{\diamond}$**90**	$*^{\diamond}$**0.00000**	* 4.77	84.41
	EHC5	$*^{\diamond}$**97**	$*^{\diamond}$**0.00000**	* **4.31**	68.87
	EHC10	$*^{\diamond}$**99**	$*^{\diamond}$**0.00000**	†* **3.19**	76.08
Predator Prey 1	MR	0	2.42447	n/a	†‡***0.01**
	Base	0	$*^{\diamond}$**0.17383**	16.28	76.36
	Ep0	0	$^{\diamond}$0.19041	†‡ 10.89	66.30
	Ep1M	0	$^{\diamond}$0.19159	†‡ 10.89	84.51
	EHC1	0	$*^{\diamond}$**0.18286**	‡ 12.61	62.70
	EHC5	0	†$*^{\diamond}$**0.16950**	13.37	71.86
	EHC10	0	†$*^{\diamond}$**0.16023**	13.54	79.33
Predator Prey 2	MR	0	0.31338	n/a	†‡***0.01**
	Base	0	†$*^{\diamond}$**0.20065**	16.72	86.67
	Ep0	0	$^{\diamond}$0.23531	†‡ 11.41	90.11
	Ep1M	0	$^{\diamond}$0.23249	†‡ 11.38	†‡29.60
	EHC1	0	$*^{\diamond}$**0.20694**	‡ 12.86	84.19
	EHC5	1	†$*^{\diamond}$**0.19156**	13.69	85.89
	EHC10	0	†$*^{\diamond}$**0.19136**	13.78	52.34
Shear Flow 1	MR	0	3.33019	n/a	0.01
	Base	$^{\diamond}$100	$^{\diamond}$0.00000	†‡1.42	9.50
	Ep0	$^{\diamond}$100	$^{\diamond}$0.00000	†‡1.35	4.00
	Ep1M	$^{\diamond}$100	$^{\diamond}$0.00000	†‡1.16	4.43
	EHC1	$^{\diamond}$100	$^{\diamond}$0.00000	‡1.83	2.70
	EHC5	$^{\diamond}$100	$^{\diamond}$0.00000	2.17	2.00
	EHC10	$^{\diamond}$100	$^{\diamond}$0.00000	†‡1.11	3.43
Shear Flow 2	MR	0	0.56694	n/a	†‡***0.01**
	Base	0	$^{\diamond}$0.00100	16.68	***32.31**
	Ep0	1	$^{\diamond}$0.00111	†‡ 10.94	104.38
	Ep1M	0	$^{\diamond}$0.00099	†‡ 11.00	106.03
	EHC1	0	$*^{\diamond}$**0.00092**	‡ 11.96	76.78
	EHC5	1	$*^{\diamond}$**0.00093**	12.32	83.49
	EHC10	0	$*^{\diamond}$**0.00040**	12.41	*90.77
van der Pol 1	MR	0	8.90324	n/a	†‡***0.01**
	Base	0	$*^{\diamond}$0.23003	18.81	87.86
	Ep0	0	$^{\diamond}$0.24255	†‡ 13.87	82.28
	Ep1M	0	$^{\diamond}$0.27929	†‡ 13.95	78.77
	EHC1	1	$*^{\diamond}$**0.20830**	14.84	85.48
	EHC5	3	†$*^{\diamond}$**0.10080**	14.95	69.16
	EHC10	0	†$*^{\diamond}$**0.10159**	15.19	66.99
van der Pol 2	MR	100	0.00000	n/a	0.01
	Base	100	0.00000	†*1.31	4.21
	Ep0	100	0.00000	2.02	1.66
	Ep1M	100	0.00000	**1.51**	2.35
	EHC1	100	0.00000	**1.54**	2.28
	EHC5	100	0.00000	†***1.34**	3.34
	EHC10	100	0.00000	2.30	4.78

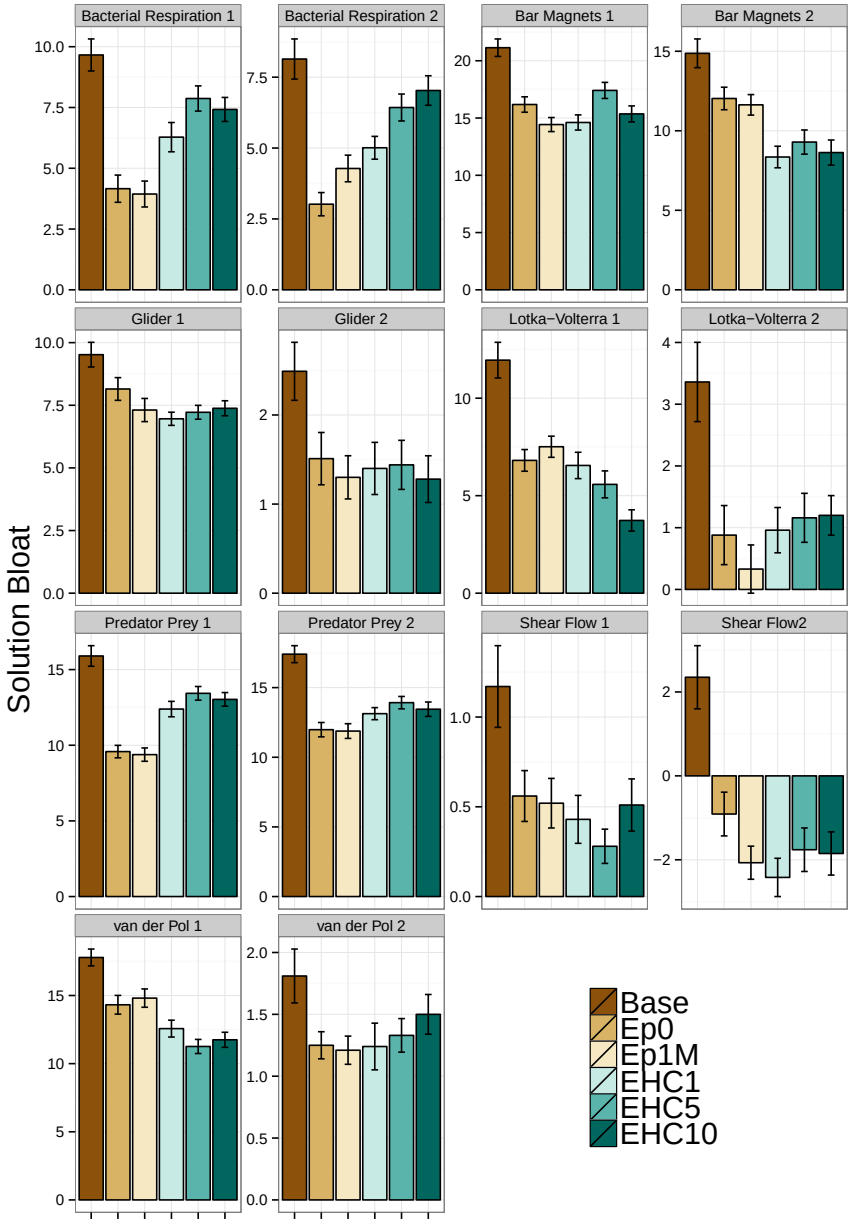

Figure 3.4. Comparison of solution bloat for the textbook ODE problems.

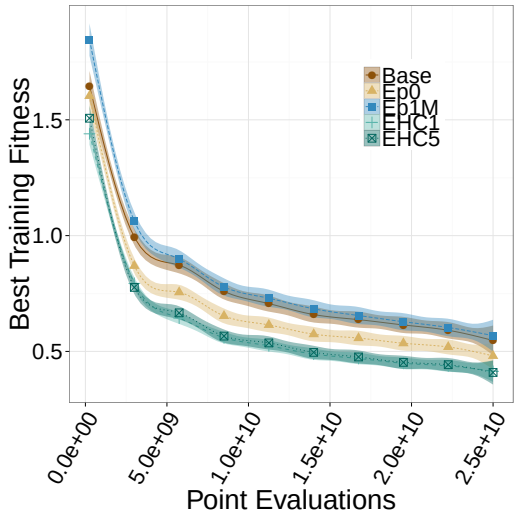

Figure 3.5. Fitness on the training set for the ODE suite.

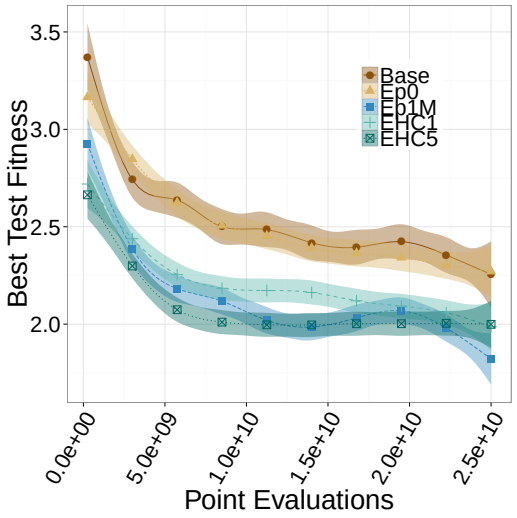

Figure 3.6. Fitness on the test set for the ODE suite.

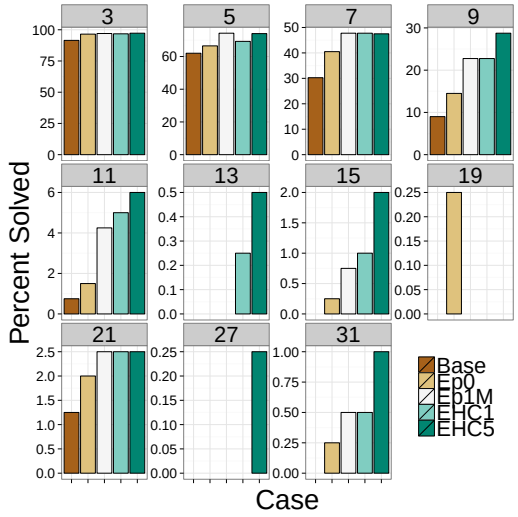

Figure 3.7. Percent of solutions found for the ODE suite. Results are grouped based on the number of nodes in the target equation (labelled at the top).

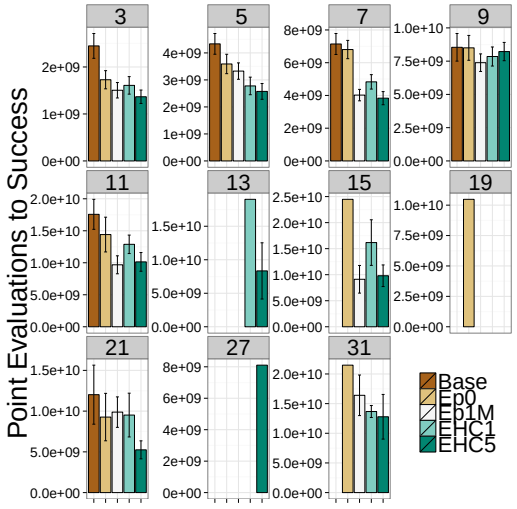

Figure 3.8. Point evaluations to success for the ODE suite. Results are grouped based on the number of nodes in the target equation (labelled at the top).

In comparison, `EHC5` applied to the measured data yields

$$\begin{aligned} \dot{\hat{y}}_1 &= -\hat{\theta}_1\sqrt{\hat{y}_1} + \hat{\theta}_1\sqrt{\hat{y}_2} \\ \dot{\hat{y}}_2 &= -\hat{\theta}_2\sqrt{\hat{y}_1} + \hat{\theta}_2(u(t) + \hat{\theta}_3) \end{aligned} \tag{3.14}$$

where $\hat{\theta}_1 = 0.0905$, $\hat{\theta}_2 = 0.0302$, and $\hat{\theta}_3 = 0.6845$. The model in Eq. (3.14) is an interesting permutation of the theoretical model (Eq. (3.13)) in that it correctly identifies the square root nonlinearities of the water levels and is as succinct as the theoretical model. However Eq. (3.14) incorrectly associates the pump input $u(t)$ and top tank level $y_1(t)$ with the second state derivative, and uses the theoretical form of $\hat{\dot{y}}_2(t)$ in Eq. (3.13) for the first state derivative $\hat{\dot{y}}_1$ in Eq. (3.14). In comparison to the theoretical model, however, the ELGP solution produces much better predictions, as illustrated by the time series comparison in Figure 3.9.

For this system, we compare the test set accuracy of the `EHC5` model to some state-of-the-art black-box models in Table 3.6, in terms of the number of parameters in the resultant model, and the mean square error (MSE) and R^2 (Eq. (3.4)) of simulated outputs on the test set. The most accurate model predictions are generated by NARX-NN, both in terms of MSE and R^2, followed by ELGP, which ties NARX-NN in prediction correlation for $y_2(t)$. ARX is the next most accurate on average, followed by the theoretical model and NARX-W. The inaccuracy of NARX-W is surprising given its complexity (64 parameters), and suggests a mismatch between the assumed nonlinearities of the approach and those present in the measured system. The NARX-NN model's excellent predictions come at the expense of complexity: the model consists of two networks with 10 hidden layers each, totaling 60 learned parameters per model. In this sense the ELGP's solution is quite reassuring because it achieves reasonable accuracy in prediction with similar complexity to the theoretical model.

The difficulty in estimating the correct model form for this system (Eq. (3.13)) may stem from the similarity of the measured outputs $y_1(t)$, $y_2(t)$, as is shown in Figure 3.9, as well as the effects of system noise and measurement error ($e_1(t)$, $e_1(t)$, $w_1(t)$, $w_2(t)$). It is especially clear from Figure 3.9 that the measured $y_2(t)$ deviates substantially from its

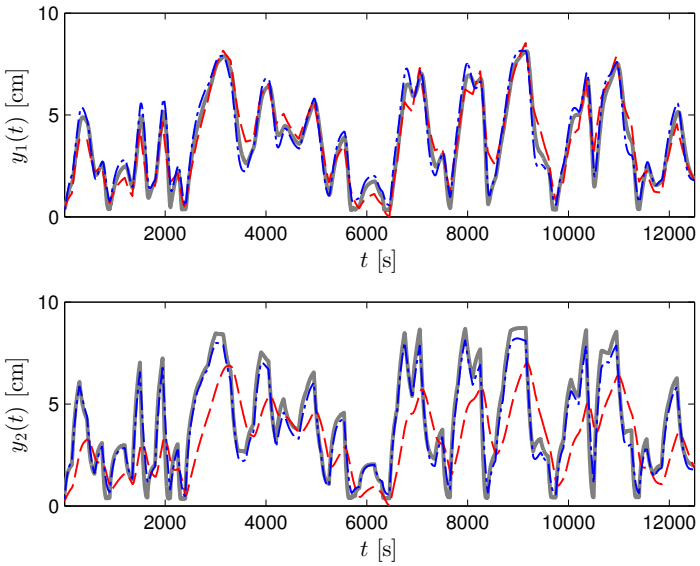

Figure 3.9. Comparison of outputs for the cascaded tanks problem, including measurement data (gray), the theoretical model (Eq. (3.13), red), and the ELGP model (Eq. (3.14), blue).

theoretical behavior, making the theoretical model difficult to infer. More fundamentally, the identification difficulty could stem from an intrinsic difficulty of ELGP in handling processes of this form subject to the measured input conditions. To determine whether or not the intrinsic difficulty of this system affects ELGP's results, we simulate the theoretical system given in Eq. (3.13) using the measured input and use the resulting outputs ($\hat{y}_1(t)$, $\hat{y}_2(t)$, $u(t)$) to train models using `EHC5`. In this case, `EHC5` renders a nearly perfect model of the theoretical system:

$$\begin{aligned} \dot{\hat{y}}_1 &= -\hat{\theta}_1\sqrt{(\hat{\theta}_2\hat{y}_1)} + \hat{\theta}_1 u(t) \\ \dot{\hat{y}}_2 &= -\hat{\theta}_3\sqrt{\hat{y}_1} + \hat{\theta}_3\sqrt{(\hat{y}_2/\hat{\theta}_4)} \end{aligned} \tag{3.15}$$

with $\hat{\theta}_1 = 0.0188$, $\hat{\theta}_2 = 0.4200$, $\hat{\theta}_3 = 0.0452$, and $\hat{\theta}_4 = 0.8830$. Both states in Eq. (3.15) have an MSE $< 10^{-6}$ and $R^2 = 1$. Thus the mismatch between the ELGP model form and the assumed process physics appears to arise from system noise and measurement error.

3.7.4 Population Diversity

Results on the textbook ODEs and the ODE suite suggest that the EHC methods improve GP performance significantly. We hypothesize that this improvement is created by preserving

Table 3.6. Mean square error (MSE) and R^2 values on the test sets for the cascading tanks problem using several modeling approaches.

Method	Parameters	MSE (test)		R^2 (test)	
		$y_1(t)$	$y_2(t)$	$y_1(t)$	$y_2(t)$
Theoretical	3	0.463	5.343	0.907	0.313
ARX	6	0.432	0.350	0.919	0.961
ELGP (EHC5)	3	0.249	0.288	0.953	0.974
NLARX-W	64	2.960	6.165	0.807	0.748
NLARX-NN	60	0.120	0.182	0.977	0.974

sections of the genome from fitness pressure and allowing them to drift genetically, thus providing an avenue for introduction of diversity and continued progress towards the solution. The syntactic and semantic similarity of models in the population can be examined in detail to determine whether this phenomenon of preserved diversity is evident. We define syntactic similarity as the homology H of a population using a Levenshtein distance comparison of S randomly sampled pairs of individuals ($|\mathbf{i}_j, \mathbf{i}_m|_L$) normalized by the length ($|\cdot|$) of the longer individual:

$$H = 1 - \frac{1}{S}\sum_{n=1}^{S} \frac{|\mathbf{i}_j, \mathbf{i}_m|_L}{max(|\mathbf{i}_j|, |\mathbf{i}_m|)} \tag{3.16}$$

We sample H each generation for both the active and inactive portions of genomes with $S = 200$. In addition, we define the semantic, i.e. behavioral, similarity of the population, referred to as *Similar Behavior*, as the fraction of identical output vectors among models in the population. This allows us to compare what is happening genetically and epigenetically at the program level (syntax) to the behavior of the resultant models (semantics).

In general we find that *silenced* genotypes have lower homology (i.e. higher diversity) than *expressed* genotypes for every epigenetic treatment, thus demonstrating that genetic drift is indeed occurring in inactive sections of programs. For instance, we measured H for the Bacterial Respiration problem (state 2) for each treatment, the results of which are shown in Figure 3.10. Homology in the expressed genome is more or less equivalent for all treatments. Despite having similar expressed genetic homologies, we find that the behavioral similarity increases more quickly with the hill climbing methods (`EHC1`, `EHC5`, `EHC10`) than with `Base`, `Ep0` or `Ep1M`, as shown in Figure 3.11. To understand why the

EHC methods converge the quickest semantically, recall that these methods only preserve epigenetic mutations that improve semantics, and are therefore more greedy than the other methods. Since the best methods for this problem converge on *Similar Behavior* the quickest, it appears that greedy topological search afforded by the EHC methods is an important factor in creating the improvements noted in our experiments. From the perspective of search, Figures 3.10 and 3.11 imply that the EHC systems are exploiting neutral variation in the genome and improved reachability in the genotype-phenotype mapping provided by epigenetics since H remains flat while *Similar Behavior* increases. Neutral variation is a property known to benefit other GP methods as well [204]. In other words, the epigenetic systems converge on model behavior more quickly while preserving genetic diversity in the search space.

It is important to note that the smoothness of the fitness landscape of a problem will play a role in determining whether greedy methods like EHC are the best option. For example, we studied several program synthesis problems for which `Ep1M` provided better performance [107]. This could be due to rugged and/or deceptive fitness landscapes. Given that the EHC methods work best for the dynamic systems studied in this chapter, it is likely that the fitness landscapes are less deceptive than those for program synthesis. It is also likely that similar systems in this domain have similar properties, and therefore they should benefit from the EHC variants of ELGP as well.

3.8 Conclusions

The results suggest that epigenetic local search is a significant addition to GP. We find that epigenetic methods, especially EHC methods, outperform a baseline implementation of GP in terms of fitness minimization, exact solutions, and equation intelligibility on textbook nonlinear ODE systems and randomly generated dynamic systems. Furthermore we show in comparison to other nonlinear approaches that ELGP is able to return concise and reasonably accurate models with similar complexity to theoretical models in a real-world application. Our study of population diversity suggests that this performance improvement to GP is

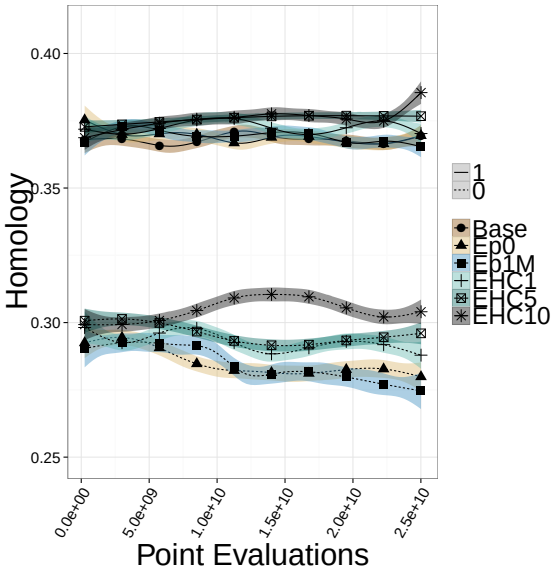

Figure 3.10. Homology among active and inactive genomes for the Bacterial Respiration 2 problem.

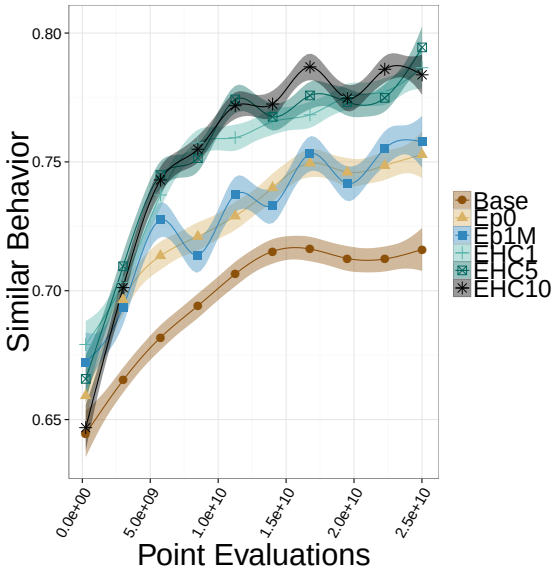

Figure 3.11. *Similar Behavior* (fraction of unique output vectors in the population) for the Bacterial Respiration 2 problem.

achieved due to the epigenetic layer's ability to preserve diversity in the inactive sequences of genes while converging more quickly in semantic space. Although we have only considered epigenetic learning by mutation and hill climbing here, the results encourage further research into the use of epigenetic methods for structure optimization in GP, and motivate a focus on methods that improve the ability of GP to search equation topologies, in addition to constants.

3.9 Acknowledgments

I thank Nicholas McPhee and the Hampshire Computational Intelligence lab for helping improve this chapter.

CHAPTER 4

MULTIDIMENSIONAL GENETIC PROGRAMMING FOR MULTICLASS CLASSIFICATION

4.1 Summary

We present a method for multiclass classification that uses genetic programming to perform a multidimensional transformation of the original attributes and then performs classification using a distance function in the transformed space[1]. The proposed classification method has the ability to optimize the extraction and synthesis of non-linear features during model development. We compare this method to several standard classification techniques across a broad set of problems and show that this technique achieves the best average test accuracy ranking while also providing dimensionality reduction and variable selection. We quantify the scalability of the method on problems of varying dimensionality and sample size. The results suggest that the added computation time is a reasonable trade-off for the increased performance on most problems.

[1]The work in this chapter is the basis of a journal publication currently under review [108].

4.2 Introduction

Classification models are a fundamental pursuit in machine learning due to their widespread utility in today's world, in applications ranging from astrophysics [198] to text classification [165] to medical diagnosis [210, 136, 149]. In multiclass classification (classification into more than two classes), we wish to find a mapping $\hat{y}(\mathbf{x}) : \mathbb{R}^p \rightarrow \mathcal{C}$ that associates the vector of attributes $\mathbf{x} \in \mathbb{R}^p$ with $k > 2$ class labels from the set $\mathcal{C} = \{c_1 \ \ldots \ c_k\}$ using n paired examples from the training set $\mathcal{T} = \{(\mathbf{x}_i, y_i), i = 1 \ \ldots \ n\}$.

Machine learning (ML) systems that conduct multiclass classification have been improved over the last 20 years [22], and open-source packages are available for performing classification (e.g., [57]) according to well-known approaches such as multiple regression (MR), support vector machine (SVM), multilayer perceptron (MLP), nearest neighbor (NN), and ensemble methods such as random forest (RF) and random subspace (RS), among others. Yet three major challenges to multiclass classification persist. The first two challenges are i) the selection of and ii) transformation of features into new features (feature synthesis), derived from the original attributes, to be used for model construction. The task of feature selection is important for reducing large-dimension data sets and for measurement selection in some domains. Typically it is left to a pre-processing step to reduce the number of attributes to a manageable size [55]; in other words feature selection is not an intrinsic property of most ML approaches. Regarding the second challenge, many ML methods employ projection of the original features into a new feature space, for example via kernel functions [143]. However the choice of kernel function is typically not automated, but picked by trial and error or cross-validation. The kernel function is important because the feature space induced by the transformation must have certain ML method-dependent properties (e.g. linear separability) in order to improve the performance of the underlying ML system. These opaque feature transformations highlight a third challenge of classification: the interpretability of the resultant models. Ideally a classifier provides insight to the user so that the classified phenomena can be better understood. This is especially relevant in the sciences and for applications like human genomics that rely on classification as a means of inferring relationships from obser-

vations. To this end, methods with intelligible representations like decision trees use greedy simplification procedures, although it is acknowledged that finding a minimal decision tree is an NP-hard problem [164].

Genetic programming (GP) [93] has been proposed for classification to remedy the three challenges above [88, 39]. GP is a stochastic optimization method that implicitly conducts feature selection by pressuring the model $\hat{y}(\mathbf{x})$ to use a subset of $\mathbf{x}$ most relevant to the problem solution. In addition, GP makes minimal *a priori* assumptions about the structure of the attribute space [116], admits a number of representations [125], and can be made to optimize the structure of the model such that it remains intelligible. Although it has been applied successfully to a number of binary classification problems [210], until recently [73, 140] it has not been competitive with standard multiclass classification techniques. The exceptions are the recently developed methods M2GP [73] and M3GP [140] that use GP to select and synthesize features and then perform classification in the new feature space using a Mahalanobis distance-based discriminant function. In this chapter, we improve upon these methods by two innovations: i) the use of a novel program representation that simplifies the construction of multidimensional representations, and ii) the incorporation of multiobjective parent selection and survival techniques that lead to more accurate classifiers. The performance of this classifier, appropriately named M4GP, is compared to that of several other methods, including M2GP and M3GP, using a set of twelve classification problems, ranging in numbers of classes, attributes and samples.

4.3 M4GP

Recall the labeled training set $\mathcal{T} = \{(\mathbf{x}_i, y_i), i = 1 \ \ldots \ n\}$, consisting of n samples of attributes $\mathbf{x}_i \in \mathbb{R}^p$ associated with the corresponding class label y_i from the set $\mathcal{C} = \{c_1 \ \ldots \ c_k\}$. The $n \times p$ matrix of attribute samples $\mathbf{X}$ can be partitioned according to its labels into k subsets $\{\mathbf{X}_1 \ \ldots \ \mathbf{X}_k\}$, such that $\mathbf{X}_j$ is the subset of $\mathbf{X}$ tagged with class label c_j. One way to classify a new sample $\mathbf{x}' \in \mathbb{R}^p$ is to measure the distance of $\mathbf{x}'$ to each subset $\{\mathbf{X}_1 \ \ldots \ \mathbf{X}_k\}$, and then assign the class label corresponding to the minimum distance [79],

i.e.

$$\hat{y}(\mathbf{x}') = c_j, \text{ if } j = \arg\min_{\ell} D(\mathbf{x}', \mathbf{X}_\ell) \ , \ \ell = 1, \ldots, k \tag{4.1}$$

One such measure is the Mahalanobis distance, D_M,

$$D_M(\mathbf{x}', \mathbf{X}_j) = \sqrt{(\mathbf{x}' - \mu_j)\, \mathbf{\Sigma}_j^{-1}\, (\mathbf{x}' - \mu_j)^T} \tag{4.2}$$

where $\mu_j \in \mathbb{R}^p$ is the centroid of $\mathbf{X}_j$ and $\mathbf{\Sigma}_j \in \mathbb{R}^{p \times p}$ is its covariance matrix, rendering D_M the equivalent Euclidean distance of $\mathbf{x}'$ from $\mathbf{X}_j$, scaled by the eigenvalues (variances) and rotated by the eigenvectors of $\mathbf{\Sigma}_j$, to account for the correlation between columns of $\mathbf{X}_j$.

This approach to classification makes several assumptions. First, each $\mathbf{X}_j$ must be sufficiently grouped such that samples always fall closest to their true distribution, which cannot be said of most difficult classification problems. Second, it assumes that the distributions of points in each $\mathbf{X}_j$ can be assumed to follow a multivariate Gaussian distribution. Third, it assumes that Eq. (4.2) can be calculated from the original data. One can imagine that as the dimensionality of $\mathbf{X}$ increases, the calculation of D_M becomes prohibitively expensive.

In order to weaken these assumptions, we wish to find a set of transformations $\Phi(\mathbf{x}) : \mathbb{R}^p \rightarrow \mathbb{R}^d$ that projects $\mathbf{x}$ into a d-dimensional space in which the samples are more easily classified according to their distribution distances. In this new space, the Mahalanobis distance takes the form $D_M(\Phi(\mathbf{x}), \Phi(\mathbf{X}_j))$, with centroid $\mu_{\Phi_j} \in \mathbb{R}^d$ and covariance matrix $\mathbf{\Sigma}_{\Phi_j} \in \mathbb{R}^{d \times d}$.

The goal of the GP system will be to find or approximate the optimal synthesized features $\Phi^* = [\phi_1 \ \ldots \ \phi_d]$ that maximize the number of correctly classified training samples, as:

$$\Phi^*(\mathbf{x}) = \arg\max_{\Phi \in \mathbb{S}} f\,(\Phi, \mathcal{T}) \tag{4.3}$$

$$f(\Phi, \mathcal{T}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\,(\hat{y}(\Phi(\mathbf{x}_i)) = y_i) \tag{4.4}$$

where $\mathbb{S}$ is the space of possible transformations Φ, f is the classification accuracy (used here as the GP fitness function), and the indicator function $\mathbb{I} = 1$ if $\hat{y}(\Phi(\mathbf{x}_i)) = y_i$, and 0 otherwise. A well-formed $\Phi(\mathbf{x})$ allows the classifier the flexibility to incorporate (linear

and/or nonlinear) transformations of the original attributes in order to improve distinctions between classes compared to using the original attribute set. By using GP to estimate the features $\Phi(\mathbf{x})$, the subset of $\mathbf{x}$ used in $\Phi(\mathbf{x})$ as well as the dimensionality of Φ, $|\Phi| = d$, are optimized. Therefore, feature selection in GP can produce $d << p$ for high dimensional data sets, making Eq. (4.2) tractable, and also admits higher-dimensional classification ($d > p$) in cases that $\mathbf{x}$ is not easily mapped to y. Note that although $D_M(\Phi(\mathbf{x}), \Phi(\mathbf{X}_j))$ assumes that the distributions of $\Phi(\mathbf{x})$ are multivariate Gaussian, this assumption is removed from $\mathbf{x}$.

4.3.1 Genetic Programming

GP solves problems by constructing and updating a population of programs composed of building blocks that represent solution components. In this case, each program consists of a set of equations that compose the synthesized features $\Phi(\mathbf{x})$ used to estimate $\hat{y}$. For example, an individual program $\mathbf{i}$ might encode the features

$$\mathbf{i} \rightarrow \Phi(\mathbf{x}) = [x_1,\ x_2,\ x_1^2,\ x_2^2,\ x_1 x_2] \tag{4.5}$$

where $\phi_1 = x_1$, $\phi_2 = x_2$, $\phi_5 = x_1 x_2$, and so on. In this case, $|\Phi| = 5$, and $\mathbf{i}$ corresponds to a polynomial expansion of two attributes.

Traditionally in GP, a program is represented by a single syntax tree evaluated based on the output generated at the root node [93]. For example, ϕ_5 above could be represented by a tree $(*\ (x_1)\ (x_2))$, where '$*$' is the root node and (x_1) and (x_2) are its leaves. However, a single output cannot represent a multi-dimensional transformation. To address this, in M2GP and M3GP, program trees were modified with special nodes in order to allow for multiple outputs at the root [73, 140]. This introduced unnecessary complexity to the representation. A contribution of this work is the introduction of a stack-based data flow to simplify the encoding of Φ, presented in §4.3.1.1.

The GP population is optimized by probabilistically selecting programs based on their performance and stochastically recombining and mutating these programs to produce a new set of programs. In this work, we implement recent selection and survival mechanisms and

compare their performance to more traditional evolutionary algorithms. These techniques are described in §4.3.1.2.

4.3.1.1 Representation

We implement a stack-based representation [156] of the equations in place of the more traditional tree-based GP representations. Programs in this representation are encoded as post-fix notation equations, e.g., $\mathbf{i} = \begin{bmatrix} x_1 & x_2 & + \end{bmatrix} \rightarrow \Phi = [x_1 + x_2]$. This representation is advantageous because it allows multiple outputs to be supported by default without the need for specialized instructions. This support is achieved by evaluating programs via executions on a stack, such that the program in Eq. (4.5) can be constructed as

$$\mathbf{i} = \begin{bmatrix} x_1 & x_2 & x_1 & x_1 & * & x_2 & x_2 & * & x_1 & x_2 & * \end{bmatrix}$$

The execution of program $\mathbf{i}$ is illustrated in Figure 4.1. Rather recursively evaluating the program as a tree starting at its root node, stack based evaluation proceeds left to right, pushing and pulling instructions to and from a single stack. Arguments such as x_1 are pushed to the stack, and operators such as '$*$' pull arguments from the stack and push the result. At the end of a program's execution, the entire stack represents the multi-dimensional transformation.

4.3.1.2 Initialization, Selection, and Variation

Programs are initialized as sets of equations varying both in individual feature size and their dimensionality. Each equation in a program is initialized recursively in an analogous fashion to the grow method (see [159]) but limited by number of nodes rather than depth. Fitness for the programs is defined in Eq. (4.4).

Three population selection methods are tested: tournament selection [42], lexicase selection [193, 60], and age-fitness Pareto optimization [178]. The first, tournament selection, is a standard GP method in which individuals (in this case, two) in the current population are randomly selected (with replacement) at a time and the one with better fitness is chosen as

$$\mathbf{i} = [\, x_1 \quad x_2 \quad x_1 \quad x_1 \quad * \quad x_2 \quad x_2 \quad * \quad x_1 \quad x_2 \quad * \,]$$
$$\text{index: } 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad 11$$

program execution	stack
1. push (x_1):	$[\, x_1 \,]$
2. push (x_2):	$[\, x_1 \quad x_2 \,]$
3. push (x_1):	$[\, x_1 \quad x_2 \quad x_1 \,]$
4. push (x_1):	$[\, x_1 \quad x_2 \quad x_1 \quad x_1 \,]$
5. pull (x_1), (x_1); push $(x_1 \cdot x_1)$	$[\, x_1 \quad x_2 \quad x_1 x_1 \,]$
6. push (x_2):	$[\, x_1 \quad x_2 \quad x_1^2 \quad x_2 \,]$
7. push (x_2):	$[\, x_1 \quad x_2 \quad x_1^2 \quad x_2 \quad x_2 \,]$
8. pull (x_2), (x_2); push $(x_2 \cdot x_2)$	$[\, x_1 \quad x_2 \quad x_1^2 \quad x_2^2 \,]$
9. push (x_1):	$[\, x_1 \quad x_2 \quad x_1^2 \quad x_2^2 \quad x_1 \,]$
10. push (x_2):	$[\, x_1 \quad x_2 \quad x_1^2 \quad x_2^2 \quad x_1 \quad x_2 \,]$
11. pull (x_1), (x_2); push $(x_1 \cdot x_2)$	$[\, x_1 \quad x_2 \quad x_1^2 \quad x_2^2 \quad x_1 x_2 \,]$

$$\rightarrow \Phi(\mathbf{x}) = [x_1,\ x_2,\ x_1^2,\ x_2^2,\ x_1 x_2]$$

Figure 4.1. Example of program representation of a multidimensional transformation. Arguments such as x_1 are pushed to the stack, and operators such as '$*$' pull arguments from the stack and push the result.

a parent for the next generation. The second, age-fitness Pareto optimization, is described in §3.6.1. Lexicase selection in more detail below.

4.3.1.2.1 Lexicase selection Lexicase selection is a new parent selection technique that rewards individuals in the population for performing well on unique combinations of fitness cases, i.e. samples. Each parent selection proceeds as follows:

1. The entire population is added to the selection pool.
2. The fitness cases are uniformly shuffled.
3. Individuals in the pool that do not have *exactly* the best fitness on the first case are removed.
4. If more than one individual remains in the pool, the first case is removed and step 2 is repeated with the next case. If only one individual remains, it is the chosen parent. If no more fitness cases are left, a parent is chosen randomly from the remaining individuals.

As evidenced above, the algorithm is quite simple to implement. In this procedure, test cases act as filters, and a randomized path through these filters is constructed each time a parent

is selected. Each parent selection event returns a parent that is elite on at least the first test case used to select it. In turn, the filtering capacity of a test case is directly proportional to its difficulty since it culls the individuals from the pool that do not solve it. Therefore selective pressure shifts to cases that are not widely solved. Because each parent is selected via a randomized ordering of test cases and these cases perform filtering proportional to their difficulty, individuals are pressured to perform well on unique combinations of test cases. This strategy promotes individuals with diverse performance, leading to increased diversity observed during evolutionary runs [60].

We use the mutation and crossover operators from M3GP [140] in order to simplify the points of comparison with M4GP. These search operators are biased to explore the dimensionality of Φ. The search operators manipulate "sub-trees" of programs, which are equivalent to segments of interacting nodes in the stack-based representation. The mutation operator, with equal probability, chooses one of three actions: i) it replaces a sub-tree with a randomly generated sub-tree; ii) it adds a new randomly generated sub-tree to the end of the program, thereby increasing $|\Phi|$ by one; iii) it deletes a sub-tree corresponding to a "root", thereby reducing $|\Phi|$ by one. The crossover operator similarly chooses between one of two equally probable actions: i) it performs standard sub-tree crossover of the parents, selecting non-"root" nodes; ii) it performs standard sub-tree crossover of "root" nodes. In this case, "roots" are those nodes in the program that produce a value in the final stack, and can be identified from stack-based programs in linear time.

4.4 Related Work

Whereas GP has been proposed for evolving classification functions $\hat{y}(\mathbf{x})$ directly [88, 39, 125], M2GP proposed GP as a wrapper that evolved $\Phi(\mathbf{x})$ for a clustering method, and demonstrated in particular that Mahalanobis distance outperformed Euclidean distance in this framework [73]. M3GP extended M2GP to allow programs to change dimensionality during the run via specialized search operators that increased or decreased the dimensionality of a tree by modifying its root node [140]. M4GP removes the need for explicit root

nodes by using a stack-based data flow that also preserves multi-dimensionality and allows dimensionality to change flexibly. An ensemble version of M3GP named eM3GP produced similar classification accuracies to M3GP with smaller, more legible resultant programs [188]. Together, these methods highlight the unique challenge of feature selection and its merger into learning systems [119].

A few recently developed ML methods have leveraged GP's feature-based abilities as a wrapper for regression [132, 72, 3]. M4GP and its ancestors differ from these regression-based approaches in that the classification does not require classes to be assigned via an arbitrarily designated range of real-valued outputs, but instead utilizes a distance metric to infer the boundaries of the transformed feature space. M4GP also incorporates a novel GP representation and advanced selection methods to improve its performance.

GP has also been proposed to fill various roles in tailored learning systems for image classification. It has been used, for example, as a way to learn image embeddings for an ensemble method [120], as an interactive learning tool for remote sensing [35], and as a binary classifier in a pulmonary nodule detection system [26]. Liu et al. [120] noted the potential for GP to perform dimensionality reduction efficiently in large-scale settings, as we noted earlier. M4GP differs from these approaches in two ways: first, it focuses on the capacity for low- and high-dimensionality feature extraction to flexibly suit the needs of the problem, and second, it applies to general multiclass classification problems.

4.5 Experimental Analysis

Ten of the twelve problems used for experimental analysis are from the UCI data repository [117] and are summarized in Table 4.1. Two others, IM-3 and IM-10, are satellite data sets from [206]. Three versions of M4GP are tested: M4GP with lexicase selection (M4GP-lx), age-fitness Pareto survival (M4GP-ps), and tournament selection (M4GP-tn). On the first 8 problems, we benchmark M4GP against M2GP, M3GP, eM3GP, and several out-of-the-box classifiers from Weka [57], including RF, RS, MLP, and SVM. Each method is run for 30 trials, and for each trial the data is randomly partitioned into 70% training and

Table 4.1. Data sets used for experimental analysis.

Data Set	Heart	IM-3	IM-10	Movl	Seg	Vowel	Wav	Yeast	Wir	Wiw	Opp-S2	Opp-S3
Classes	2	3	10	15	7	11	3	10	10	10	4	4
Attributes	13	6	6	90	19	13	40	8	12	12	242	242
Samples	270	322	6798	360	2310	990	5000	7797	1599	4898	16667	15550

30% testing. For the final four problems, we benchmark M4GP against published results for red and white wine quality [29] and human activity recognition [172, 23]. The settings for M4GP are shown in Table 4.2, and whenever possible, match those used for M2GP, M3GP, and eM3GP. Programs are constrained to be between 3 and 100 nodes, and are initialized with a corresponding dimensionality between 1 and 33, constrained such that the minimum sub-program is at least 1 node.

In order to test the scalability of M4GP, we used the Opportunity Activity Recognition data set (Opp-S2 and Opp-S3 in Table 4.1) from the Activity Recognition Challenge at the 2011 IEEE International Conference on Systems, Man, and Cybernetics [172, 23], which consists of 242 attributes and approximately 32,000 total samples. We benchmark M4GP's ability to predict four classes of locomotion (stand, sit, walk, lie) from two test subjects (S2 and S3) against the entrants to the original challenge. In order to perform the comparison to published results, the weighted F-measure, F_1, is used as the fitness metric in place of Eq. (4.4). F_1 measures classification performance as a function of precision ($\frac{TP}{TP+FP}$) and recall ($\frac{TP}{TP+FN}$) as:

$$F_1 = \sum_{\ell}^{k} 2w_\ell \frac{\text{precision}_\ell \cdot \text{recall}_\ell}{\text{precision}_\ell + \text{recall}_\ell} \tag{4.6}$$

TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives for class c_ℓ that has k_ℓ samples out of a total of n, yielding the proportion weight $w_\ell = k_\ell/n$. The raw time series data was downsampled as in [23] using a moving average with a window of 500 milliseconds with 250 millisecond steps. Missing data is linearly interpolated, and all data was normalized to zero mean and unit variance.

Table 4.2. M4GP settings. Changes to the settings for Opp-S2 and Opp-S3 are noted in parentheses.

Setting	Value
Population size	500 (1000)
Max Generations	100
Crossover / Mutation	50/50%
Ephemeral random constants [93] range	[0,1]
Program size limits by # nodes	[3, 100 (500)]
Initial dimensionality range (d)	[1,33]
Termination criterion	generations or perfect training accuracy
Trials	30 (10)

4.6 Results

The best classifiers generated by M4GP for each trial are compared first to benchmark methods in §4.6.1 and then to other published results in §4.6.2. Performance is quantified by classification accuracy (Eq. (4.4)) on test data and the dimensionality of the resultant classifiers.

4.6.1 Benchmark Comparisons

For the 8 benchmark problems, the median best fitness on the test sets for the first eight problems are shown in Table 4.3 with statistical comparisons. M4GP, across selection methods, produces the best classifiers in terms of test accuracy on five of the eight problems, and RF produces the best classifiers on the remaining three. The best-of-run classifier accuracies over all trials for all methods on the test sets are plotted in Figure 4.2. Interestingly, the M4GP results are not typically best on the training data, but are most often the best on the test sets, signifying that the classifiers produced by M4GP generalize better than the other tested methods. The results are summarized by rankings over all of the problems on training and test sets in Figures 4.4 and 4.5. Again these two figures illustrate the improved generalization achieved by M4GP, which ranks first across selection methods for test accuracy but not for training accuracy. The performance of the three M4GP methods on training and test sets over the duration of the run (generations) is shown in Figure 4.3. It is worth noting that for the Movl problem, which is one of the three problems for which RF generalizes better, M4GP finds a perfect solution to the training data within the first few generations, and pre-

maturely terminates. Past work also notes that this problem is an outlier in terms of GP's behavior [73, 140]. On the Heart problem, M4GP-lx overfits to the training data, as evidenced by the slow decline in test fitness over the generations. Among selection methods, M4GP-ps ranks the best, followed by M4GP-lx and M4GP-tn. On individual problems, M4GP-tn did not outperform M4GP-ps or M4GP-lx; therefore, M4GP-tn was left out of further tests.

4.6.2 Comparison to Published Results

The wine results are compared to published results [29] in Table 4.4. The classifiers generated by M4GP, both using lexicase selection and Pareto survival, are significantly more accurate than those generated by MR and MLP for this problem. In comparison to SVM results, there is no significant difference between M4GP-ps and SVM for the red wine problem, and SVM is significantly more accurate than M4GP-lx and M4GP-ps for the white wine problem. The white wine problem is the only problem studied for which an SVM approach achieves a better classification accuracy than M4GP.

The results of the Opportunity Activity Recognition problem are shown in Table 4.5 where the results of M4GP are compared to those from the original challenge in terms of F-measure (Eq. (7.2)). A number of methods are reported for this problem, including nearest neighbor (NN) classifiers, SVM, decision tree and ensemble versions thereof. For both subjects tested, M4GP-lx and M4GP-ps produce better classifiers than the competition, with lexicase selection performing slightly better than Pareto survival. Because only single best results are reported in literature, we are unable to provide statistical tests for these results. However, we report the median F-measure on the test set for 10 trials of M4GP-lx and M4GP-ps along with confidence bounds, indicating that the median performance of M4GP exceeds the best reported result from the competition.

4.6.3 Scalability

A motivating factor for using M4GP for classification is its intrinsic feature selection capacity. The complexity of the M4GP solutions are detailed in Figure 4.6, where the number of original attributes used in solutions as well as the dimensionality of the transformation (i.e.

$|\Phi|$) is shown for all the problems. For reference, we include a principal components analysis (PCA) reduction that shows the fewest principal components needed to account for 98% of the variance of each data set. For some of the problems with small numbers of attributes (e.g. Yeast, IM-10 and Heart) the dimensionality of the M4GP solutions tend to be larger than the dimensionality of the original attribute set, but for higher-dimensional problems (e.g. Opp-S2 and Opp-S3), large reductions in the dimensionality of the problem are achieved via M4GP that surpass the reduction given by PCA. It is also important to note that often the M4GP solutions make use of fewer of the original attributes, which can be beneficial for sensor or measurement selection in future design of experiments.

As a population-based method, M4GP's main drawback in comparison to other methods is computational cost, shown in terms of wall-clock time in Table 4.6. Each trial of the initial ten problems from Table 4.1 were run on a single core, so the reported time is the time to evaluate population solutions in series. The times range from about 30 seconds for simple problems (e.g. Movl) to about two and a half hours for larger sample size problems (e.g. IM-10). The largest problems, Opp-S2 and Opp-S3, that have close to three times as many training samples as IM-10, were run on 16 core machines, which significantly reduced the run-time to around 10 minutes. In this case, the problem is run in parallel using an island model [8], in which the population is divided among the cores. The resulting computation times indicate the improvement afforded by parallel processing in GP methods.

4.7 Discussion and Conclusion

Across the range of problems studied, the results indicate that M4GP shows promise as one of, if not the, best method for developing accurate classifiers that generalize well. In addition to being accurate, M4GP provides the flexibility to increase dimensionality for better classification of small dimension problems, and to decrease both the dimension of solutions and the number of necessary attributes for large dimension problems. For the Opp-S2 and Opp-S3 problems in particular, the dimensionality of the classifiers produced by

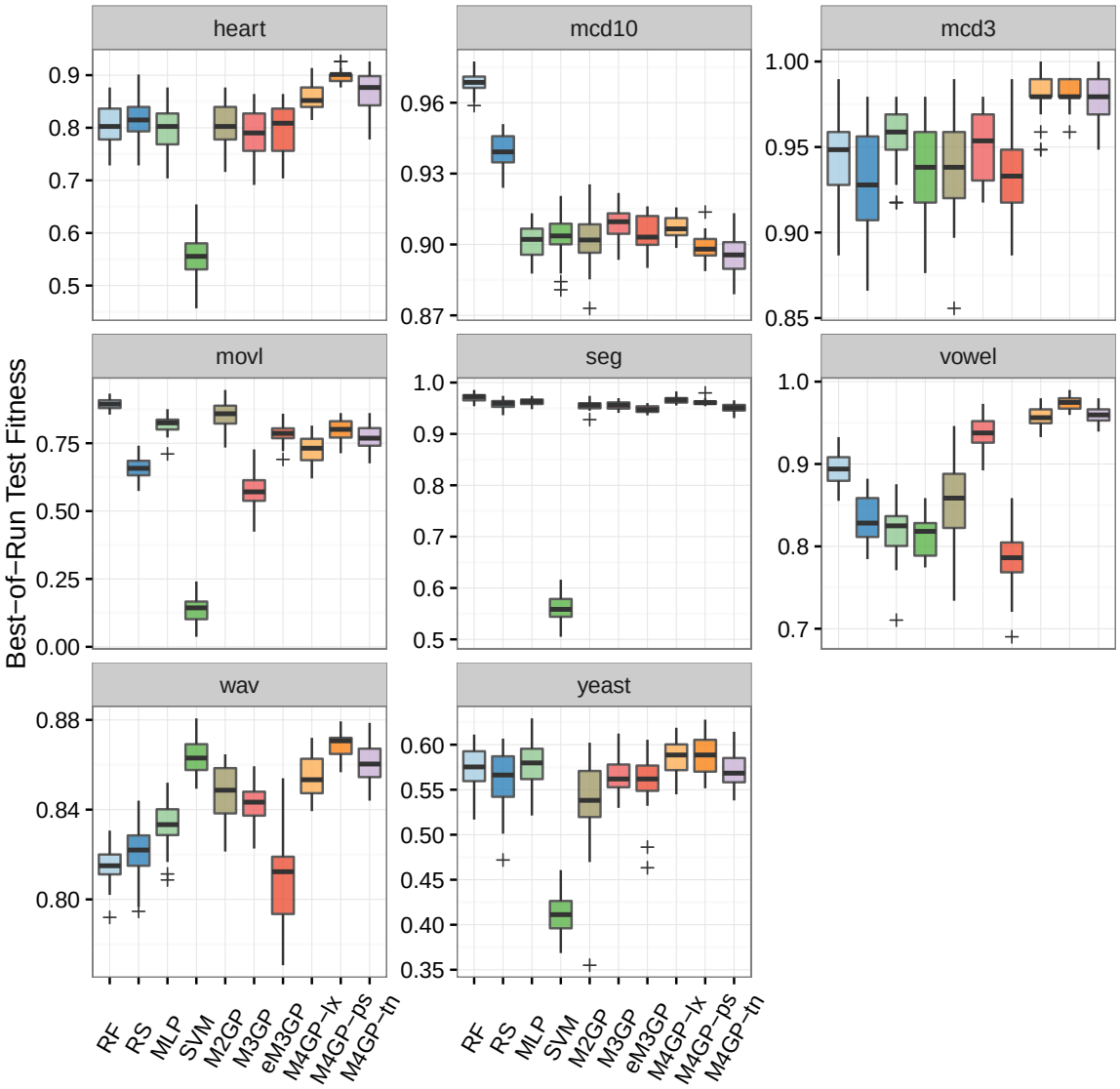

Figure 4.2. Test set accuracy on the first eight benchmark problems for ten different classification methods.

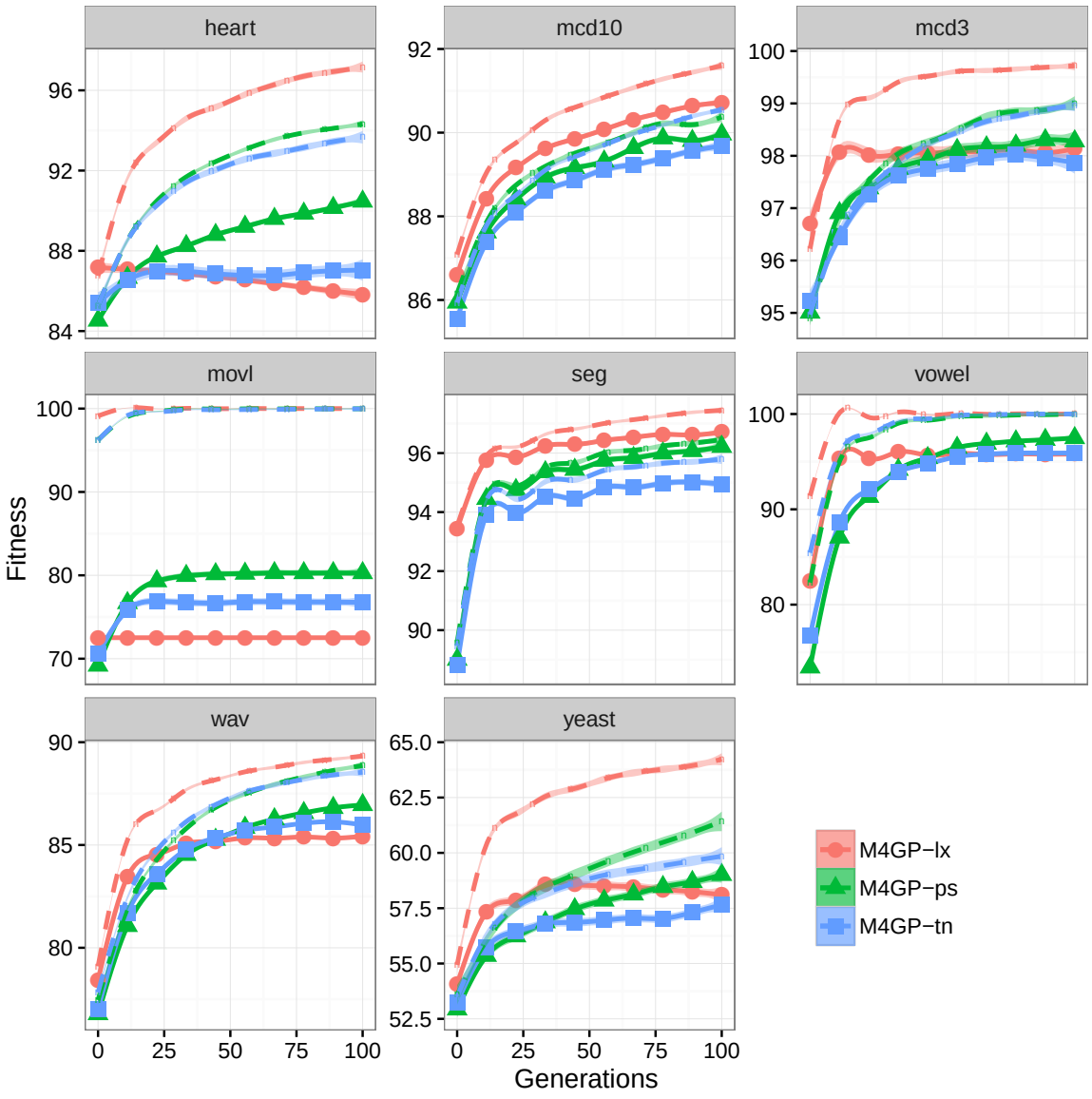

Figure 4.3. Convergence characteristics of M4GP on the training set (dotted lines) and test set (shapes) over 100 generations.

Table 4.3. Comparison of best-of-run median test accuracy for the benchmark problems. The best result is highlighted. Significant ($p < 0.01$ according to a pairwise Wilcoxon rank-sum test with Holm correction) improvements with respect to each method is denoted by $a - j$ according to the method labels.

Method	Heart	IM-3	IM-10	Movl	Seg	Vowel	Wav	Yeast
[a]RF	[d]80.2	94.8	[bcde fghij]96.9	[bcde fghij]89.4	[bcde fgij]97.3	[bcde g]89.4	81.5	[de]57.5
[b]RS	[d]81.5	92.8	[cde fghij]93.9	[d f]65.7	[d gj]96.0	[g]82.8	82.2	[d]56.6
[c]MLP	[d]80.2	95.9	90.2	[bd fghj]82.5	[de gj]96.3	[g]82.5	[ab g]83.3	[de]58.0
[d]SVM	55.6	93.8	90.4	14.4	55.8	81.8	[abce fgh]86.3	41.1
[e]M2GP	[d]80.2	93.8	90.2	[bcd fghij]85.9	[d g]95.6	[cd g]85.9	[abc g]84.9	[d]53.8
[f]M3GP	[d]79.0	95.4	[c ij]91.0	[d]57.1	[d]95.6	[abcde g]93.8	[ab g]84.3	[d]56.2
[g]eM3GP	[d]80.9	93.3	[j]90.3	[bd fh]78.6	[d]94.7	78.6	81.2	[d]56.2
[h]M4GP-lx	[abcde fg]85.2	[abcde fg]97.9	[ij]90.7	[bd f]73.1	[bde fgj]96.6	[abcde fg]95.6	[abc fg]85.3	[de g]58.9
[i]M4GP-ps	[abcde fghj]90.1	[abcde fg]97.9	89.8	[bd fh]80.1	[de gj]96.1	[abcde fghj]97.5	[abce fghj]87.1	[de fg]58.9
[j]M4GP-tn	[abcde fg]87.7	[abcde fg]97.9	89.6	[bd fh]76.9	[d]95.1	[abcde fg]96.0	[abce fg]86.0	[d]56.8

Table 4.4. Comparison of mean test accuracy on the wine problems. M4GP is compared to published results using multiple regression (MR), mulitlayer perceptron (MLP) and support vector machines (SVM). The best results are highlighted. Significant ($p < 0.01$ according to a pairwise t-test) improvements with respect to each method is denoted ($a - e$) according to the method labels (left).

Method	Red Wine	White Wine
[a]MR	59.1±0.1	51.7±0.1
[b]MLP	59.1±0.3	[a]52.6±0.3
[c]SVM	[ab d]62.4±0.4	[ab de]64.6±0.4
[d]M4GP-lx	[ab]60.8±0.2	[a]53.3±0.1
[e]M4GP-ps	[ab]61.2±0.2	[ab d]54.2±0.1

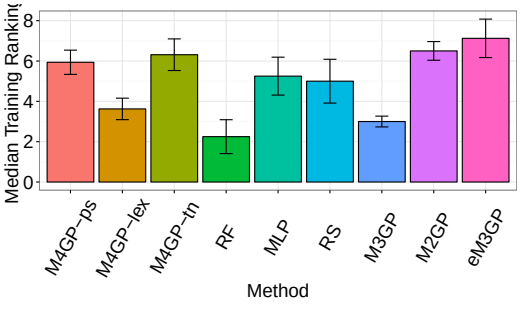

Figure 4.4. Mean training rankings of classifiers on the eight benchmark problems. Bars denote the 95% confidence interval.

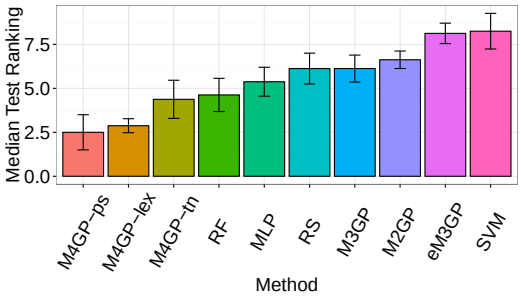

Figure 4.5. Mean test rankings of classifiers on the eight benchmark problems. Bars denote the 95% confidence interval.

Table 4.5. Comparison of F-measure on the Opportunity Activity Recognition data set (locomotion) for subjects 1 and 2 (S1 and S2). M4GP is compared to published results using one nearest neighbor (1-NN), SVM, SVM + 1-NN, decision trees (C4.5), k-NN, decision tree (DT) grafting, and Adaboost. The team names are shown in parentheses. The best results in terms of F-measure are highlighted. For M4GP results, the median best result of ten trials is shown with the 95% confidence interval.

Method	F-measure (no *Null* class)	
	S2	S3
1-NN (NStar)	0.88	0.85
SVM (SStar)	0.87	0.83
SVM + 1-NN (CStar)	0.90	0.83
C4.5 (NU)	0.83	0.63
k-NN (MI)	0.87	0.86
DT grafting (MU)	0.86	0.87
Adaboost (UT)	0.74	0.72
M4GP-lx	0.91 ± 0.00	0.89 ± 0.00
M4GP-ps	0.90 ± 0.00	0.88 ± 0.00

Table 4.6. Run time for M4GP solutions.

		Median Time (hr:min:s)		
Problem	Cores	M4GP-lx	M4GP-ps	M4GP-tn
Heart	1	00:02:29	00:02:12	00:02:28
IM-3	1	00:02:30	00:02:49	00:02:14
IM-10	1	02:09:04	02:21:45	02:28:23
Movl	1	00:00:27	00:02:25	00:02:02
Seg	1	00:32:25	00:39:35	00:38:38
Vowel	1	00:04:21	00:20:37	00:15:57
Wav	1	01:36:44	01:30:17	01:35:12
Yeast	1	00:25:25	00:44:09	00:52:31
Red Wine	1	00:20:36	00:11:14	- (-)
White Wine	1	00:56:26	00:44:12	- (-)
Opp S2	16	00:12:30	00:09:16	- (-)
Opp S3	16	00:11:51	00:08:47	- (-)

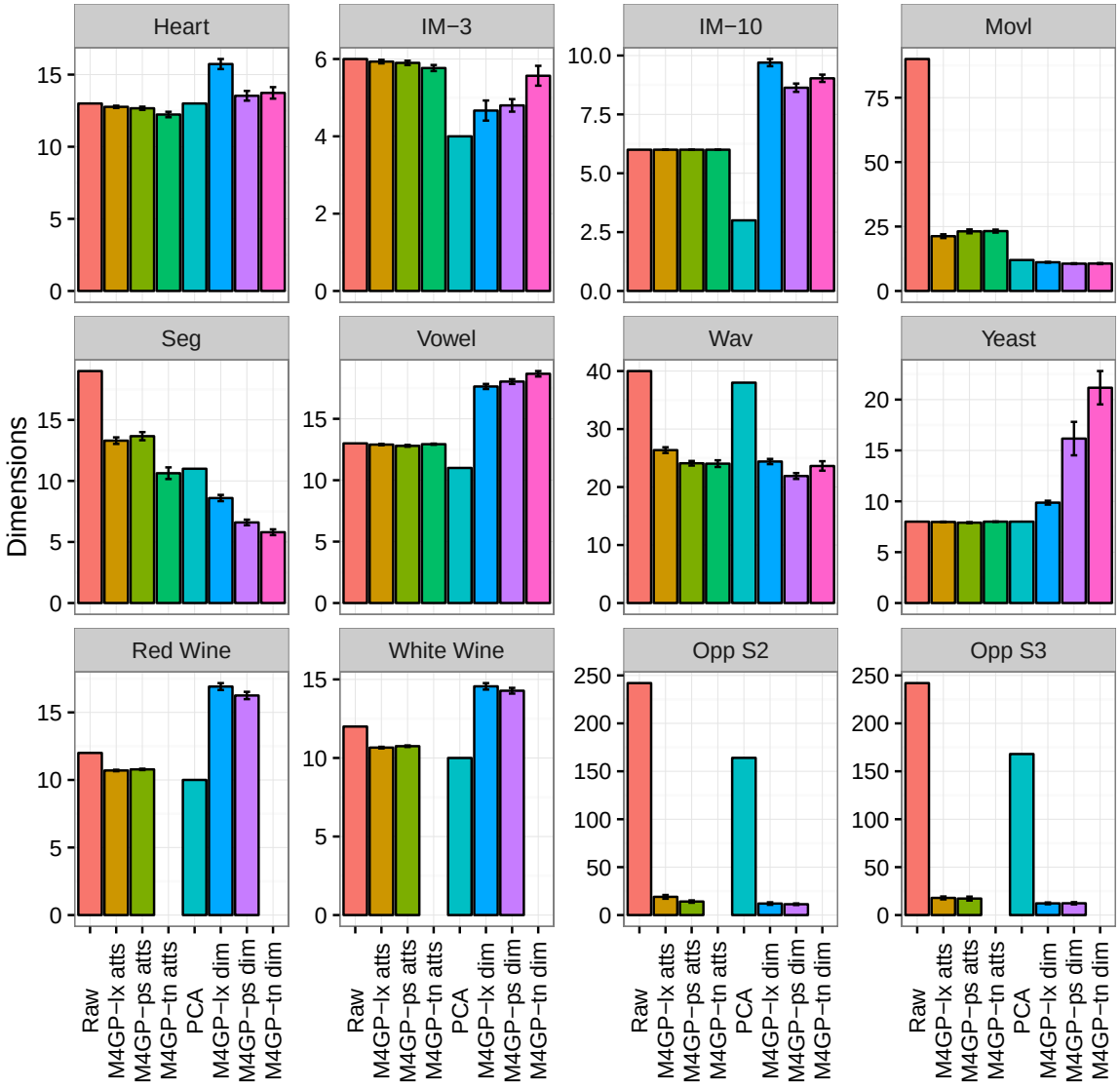

Figure 4.6. Dimensionality reductions afforded by M4GP compared to the original number of attributes and a PCA dimensionality reduction preserving 98% variance. 'atts' refers to the number of attributes used in the M4GP solutions, and 'dim' refers to the dimensions in the solution (i.e. $|\Phi|$).

M4GP constitute a major reduction from the original attribute space beyond that afforded by PCA reduction and motivate further applications of this method to large attribute problems.

We find that the computation time of M4GP is quite reasonable on a multi-core machine, with the largest sample size problems requiring approximately 10 minutes to run on a 16 core machine. Improvements to computation time could be made by further parallelizing the execution of individual models that can be run in parallel at the data (samples) and program (nodes) levels. Note that in this work we only parallelize the evaluation of the population of models. The largest computational cost stems from the inversion of the covariance matrix to compute the Mahalanobis distance (Eq. (4.2)). It may be possible to minimize this cost by only maintaining a single model Φ and evolving a population that corresponds to the dimensions of Φ, thereby minimizing the number of matrix inversions to once per generation. Our future work could include this direction. Furthermore, this work motivates further research into pairing other classifiers with GP feature selection and synthesis.

Finally, we demonstrate that M4GP significantly improves upon previous versions of Mahalanobis distance-based GP embodied by M2GP, M3GP, and eM3GP. These improvements are brought about by replacing the tree-based programming representation with a stack-based data flow, thereby easing the creation of and search of multi-dimensional classification models. M4GP performs significantly better than these previous methods across all benchmark problems according to a Friedman test of rankings for test accuracy. We further demonstrate that advanced evolutionary selection methods, namely lexicase selection and age-fitness Pareto survival, perform better than tournament selection in producing accurate classifiers.

4.8 Acknowledgments

The authors would like to thank Mauro Castelli for his feedback as well as members of the Computational Intelligence Laboratory at Hampshire College. This work is partially supported by the NSF-sponsored IGERT: Offshore Wind Energy Engineering, Environmental Science, and Policy (Grant Number 1068864), as well as Grant No. 1017817,

1129139, and 1331283. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1053575[202]. We acknowledge the financial support of BioISI R&D unit, UID/MULTI/04046/2013 funded by FCT/MCTES/PIDDAC, Portugal.

PART II: APPLICATIONS

CHAPTER 5

RESTRUCTURING CONTROLLERS TO ACCOMMODATE PLANT NONLINEARITIES

5.1 Summary

The possibility of controller restructuring using MSAM is explored for improved closed-loop control of nonlinear plants. The starting controller in this restructuring approach can be the linear controller designed according to the linearized model of the plant. This controller will be expanded into a set of nonlinear candidate controllers to be adapted iteratively toward delivering a desired closed-loop response. The best candidate controller that betters the initial controller's performance is selected for further adaptation. The salient feature of the proposed adaptation method is a metric for quantifying structural perturbations to the controllers, which it uses for scaling the structural Jacobian that is central to its gradient-based mechanism. Results obtained from two case studies indicate the success of the proposed restructuring approach in finding nonlinear controllers with improved closed-loop response and higher robustness to modeling uncertainty.

5.2 Introduction

When the agility of feedback can account for mild plant nonlinearities, linear controllers can be designed according to the linearized model of the plant [47]. And in cases when the plant nonlinearities are too severe for a single linear controller across the range of operating points, gain scheduling can be employed to incorporate different linear controllers at different operating points [4]. The leap to nonlinear control is made when accurate models of plant nonlinearities exist to allow nonlinear controller design [100, 183, 86]. This paper offers an alternative method of empirical controller development wherein a starting controller is expanded into a nonlinear controller and adapted to attain improved closed-loop performance.

The most common framework for empirical development of nonlinear controllers has been neural networks [145, 146, 214, 67]. However, these controllers have a "black box" form which precludes any analysis requiring intelligible forms and/or structures. In an attempt to attain transparency, one can use symbolic regression wherein the process variables, inputs, and parameters (constants) are treated as symbols and integrated as blocks to form candidate models. Free of restrictions on the form (structure) of candidate controllers, the search would be conducted by genetic programming (GP) for controllers generating best-fit closed-loop outputs to the desired response [92]. But symbolic regression is computationally expensive, requiring anywhere from thousands to billions of evaluations. While so many evaluations can be accommodated in open-loop by algebraic manipulation of the time series representing measured observations and their derivatives, they are infeasible in closed-loop wherein the system response needs to be obtained via simulation for each adopted controller. As such, the use of evolutionary and/or genetic algorithms in controls has been confined to parameter optimization [44, 166] or search among a limited number of structural components [25, 10].

The approach proposed in this chapter also restricts the search space to a limited number of candidate controller structures. However, instead of fixed structures to select from, it performs a local search around the starting controller. It considers candidate controllers derived from a starting controller that are malleable through their exponents and can be adapted by gradient-based search toward a suitable form. These candidate controllers are obtained

by restructuring the individual components of a starting controller by the Model Structure Adaptation Method (MSAM) [111], described in Ch. 2, which adds variable coupling to individual components of the starting controller and tunes their influence by gradient-based adaptation of their exponents. A key feature of MSAM, that enables the implementation of gradient-based adaptation, is its quantification of structural changes to the controllers. It uses this metric to scale the structural sensitivities such that they will remain robust to parametric error during adaptation. The proposed restructuring strategy is schematized in Fig. 5.1, which resembles the strategy used in iterative feedback tuning [62, 115, 87], with the difference that it changes the structure instead of the parameters toward the desired response. In Fig. 5.1, the y^d denotes the desired response to the reference input r, and $\tilde{y}$ represents the error between the closed-loop response of the system y and its desired response y^d.

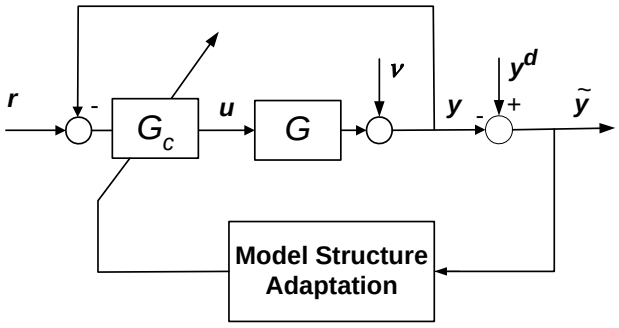

Figure 5.1. Contoller adaptation by MSAM

5.3 MSAM for Controller Adaptation

Whereas MSAM begins with starting model in system identification applications (see Ch. 2), here it begins with a starting controller. It uses the components of this starting controller and creates pair-wise coupling between the individual components, amended by exponents. It then forms a set of restructured candidate controllers from different combinations of coupled components and adapts the exponents of each controller by gradient-based search to optimize the influence of individual couplings. It subsequently evaluates the perfor-

mance of these candidate controllers in closed-loop to find the best controller that surpasses the starting controller in matching the desired response y^d. The salient feature of MSAM is its use of a metric for symbolic changes to the model, which it uses for scaling the structural Jacobian. This scaling is shown to improve the condition of the structural Jacobian, for reliable implementation of gradient-based adaptation in the symbolic domain.

In MSAM, the starting controller is considered to be the weighted sum of individual components Ψ_i, as

$$\mathbf{M}_{\boldsymbol{\Theta}} = \sum_{i=1}^{Q} \theta_i \Psi_i = \boldsymbol{\Theta}^T \boldsymbol{\Psi} \tag{5.1}$$

where $\boldsymbol{\Psi} = [\Psi_1, \ldots, \Psi_Q]^T$ comprises components Ψ_i that are products of combinations of state variables x_i included in the state vector $\mathbf{x} = [x_1, \ldots, x_n]^T$. For instance, with a PID controller as the starting controller $\widetilde{\mathbf{M}}_{\widetilde{\boldsymbol{\Theta}}} : u(t) = K_p\epsilon(t) + K_i \int \epsilon(t)\mathrm{d}t + K_d \mathrm{d}\epsilon/\mathrm{d}t$, $\widetilde{\boldsymbol{\Psi}} = \left[\tilde{\Psi}_1,\ \tilde{\Psi}_2,\ \tilde{\Psi}_3\right]^T = \left[\epsilon(t),\ \int \epsilon(t)\mathrm{d}t,\ \mathrm{d}\epsilon/\mathrm{d}t\right]^T$ with the corresponding parameter values $\widetilde{\boldsymbol{\Theta}} = \left[\tilde{\theta}_1,\ \tilde{\theta}_2,\ \tilde{\theta}_3\right]^T = [K_p,\ K_i,\ K_d]^T$. The fidelity of the controller can be evaluated by how closely the closed-loop response of the nonlinear plant matches the desired response y^d, as represented by $\tilde{y}_{\widehat{\mathbf{M}}}$ where $\widehat{\mathbf{M}}$ denotes the candidate controller.

The fitness function in MSAM (Eq. 2.18) is defined according to the simulated and desired output as

$$F = \frac{\rho(\hat{y}, y^d)}{\sum_{k=1}^{N} |\hat{y}(t_k) - y^d(t_k)|} \tag{5.2}$$

where $\rho(\hat{y}, y^d)$ denotes the correlation coefficient between the closed-loop response $\hat{y}$ and the desired response y^d. The controller form is adapted according to the procedure described in §2.5.

5.4 Study Platforms

Two closed-loop platforms are considered for studying the feasibility of MSAM. The first platform, depicted in [4], consists of a linear plant that is actuated by a nonlinear valve, thus the known plant nonlinearity is compartmentalized. Åström and Wittnemark [4] capitalize on knowledge of the valve nonlinearity to cascade the linear (proportional plus integral (PI))

controller with the inverse function of the valve model, so as to negate/compensate for its nonlinearity. The second platform is an inverted pendulum on a cart which is an inherently nonlinear and unstable system commonly controlled within small deviations from its vertical position. Effectiveness of controller restructuring is evaluated in application to these two platforms and compared to those of traditional solutions offered for their control.

5.4.1 Nonlinear Actuator Valve

The first platform adopted from [4] is shown in Fig. 5.2 where the plant consists of a nonlinear valve,

$$v = f(u) = u^4 \tag{5.3}$$

preceding a linear process having the transfer function

$$G_0(s) = \frac{1}{(s+1)^3} \tag{5.4}$$

The customized controller discussed in [4] is a PI controller with the parameters $K_p = 0.1$ and $T_i = 0.1$ cascaded with a nonlinear function that approximates the inverse of the valve model, as

$$f^{-1}(c) = \begin{cases} 0.433c & \text{if } 0 \leq \text{c} \leq 3 \\ 0.0538c + 1.139 & \text{if } 3 \leq \text{c} \leq 16 \end{cases} \tag{5.5}$$

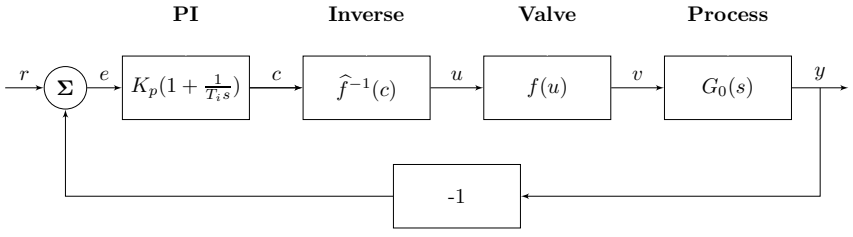

Figure 5.2. Block diagram of the closed-loop control system for the nonlinear valve with the customized controller (Courtesy of Åström and Wittnemark [4]).

As demonstrated in [4], and shown in Fig. 5.3, the above closed-loop system provides different responses at different reference values, representing the limitation of the inverse approximation $\widehat{f}^{-1}$ in neutralizing the valve nonlinearity $f(u)$ at different reference values. One drawback of this solution, therefore, is rooted in the deviation of $f(\widehat{f}^{-1}(c))$ from the ideal value of 1 at different reference values. Its another drawback is its dependence on the accuracy of the valve nonlinearity. To evaluate the significance of this dependence, the closed-loop step responses of the system at different reference values are compared with the step responses of two other systems representing slightly different valve nonlinearities: $f(u) = u^{3.5}$ and $f(u) = u^{4.5}$ in Fig. 5.4. The results clearly indicate the considerable influence of misrepresented nonlinearity on the response of the customized solution, particularly at higher reference values.

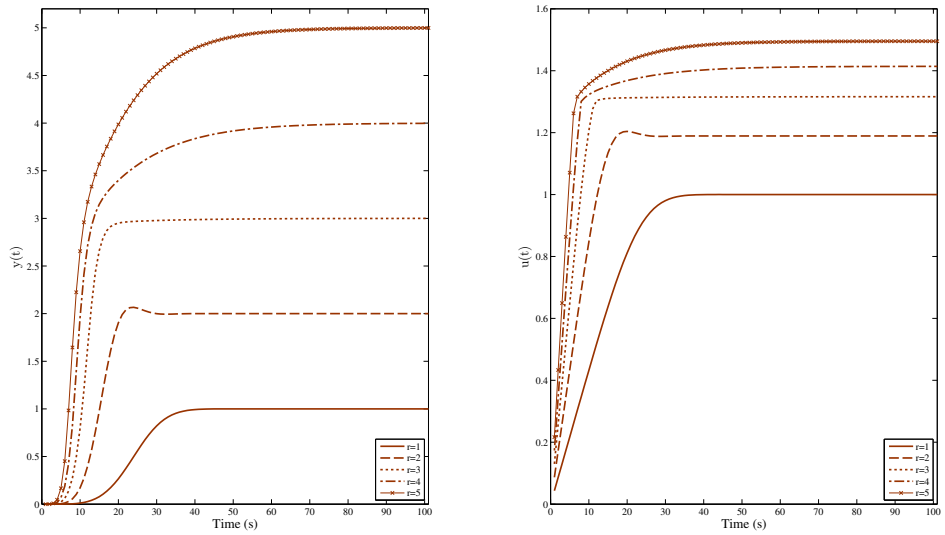

Figure 5.3. Step responses and control efforts of the closed-loop system in Fig. 5.2 at different reference values

5.4.2 Inverted Pendulum

The second platform is the classical inverted pendulum on a cart as shown in Fig. 5.5. With the cart mass denoted as M, the mass of the pendulum at the end of the massless rod represented as m, the position of cart in x-direction denoted as $x(t)$, the angle of the

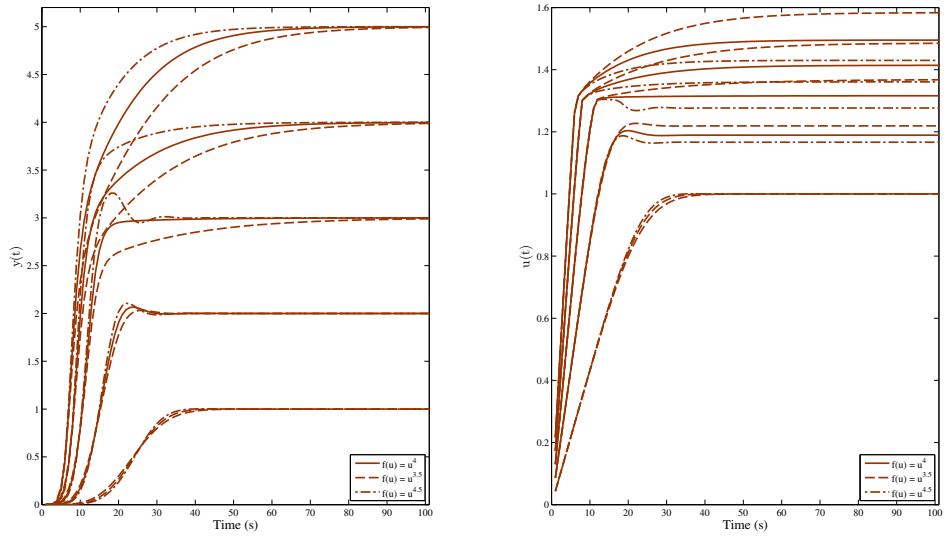

Figure 5.4. Step responses and control efforts of the closed-loop customized solution in Fig. 5.2 affected by the valve nonlinearity

pendulum from vertical represented as $\theta(t)$, and the control action being a force $u(t)$ applied to the cart, the system has the model [137]

$$\begin{aligned} \ddot{x} &= \frac{u + ml\left(\sin(\theta)\right)\dot{\theta}^2 - mg\cos(\theta)\sin(\theta)}{M + m - m\cos^2(\theta)} \\ \ddot{\theta} &= \frac{u\cos(\theta) - (M+m)g\sin(\theta) + ml\left(\cos(\theta)\sin(\theta)\right)\dot{\theta}}{ml\cos^2(\theta) - (M+m)l} \end{aligned} \tag{5.6}$$

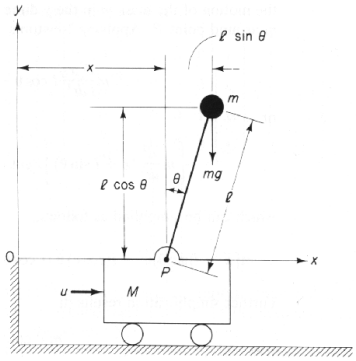

Figure 5.5. Inverted Pendulum on cart

The feature of interest in this platform is the dependence of the system nonlinearity on the angle θ. At small θ values, like those caused by low magnitude impulses to the pendulum, a linear controller, by state feedback, for example, can maintain the upward position of the pendulum. But the prevalence of nonlinearity at larger θ values will disturb the performance of linear control. This point is illustrated for a linear state-feedback controller of the form $u(t) = -K_1x - K_2\dot{x} - K_3\theta - K_4\dot{\theta}$ with the gains $[K_1, K_2, K_3, K_4] = [-2.00, -3.84, 33.84, 7.22]$ configured to yield the closed-loop poles $s_{1,2,3,4} = -1, -2, -4.73, -4.73$ according to a linearized model of the pendulum. The closed-loop impulse responses of the system to different impulse magnitudes using this controller are shown in Fig. 5.6. They clearly indicate the effect of nonlinearity on the performance of the controller at higher impulse magnitudes.

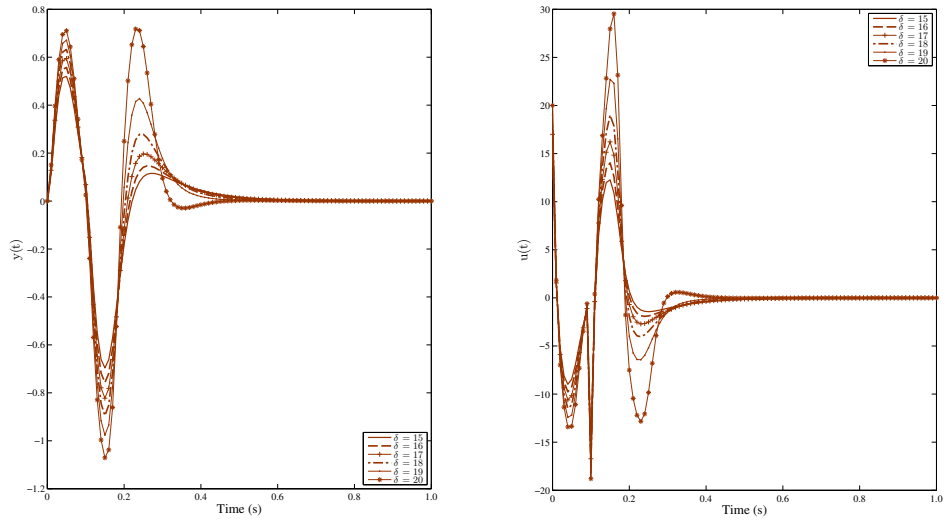

Figure 5.6. Closed-loop impulse responses and control efforts of the inverted pendulum controlled by linear state feedback

5.5 Features of MSAM-restructured controllers

Before we discuss the characteristic performance of the restructured controllers, it is necessary to note some of the features of MSAM-restructured solutions. The first is the significance of the model perturbation magnitude in the generated solution. Second is the case-specificity of the solutions.

5.5.1 Significance of model perturbation magnitude

An important feature of MSAM is the use of $\delta\Psi_i$ from Eq. (2.14) for scaling the columns of $\mathbf{\Phi}_\gamma$. A direct ramification of this scaling is the better quality of $\mathbf{\Phi}_\gamma$, that results in better estimates of $\widehat{\Delta\mathbf{\Gamma}}$ when used in Eq. (2.16). The improved quality of $\mathbf{\Phi}_\gamma$ is illustrated by the range of condition numbers ($\lambda_{max}/\lambda_{min}$) of $\mathbf{\Phi}_\gamma$ in Table 5.1, computed with and without scaling by $\delta\Psi_i$ at different reference values with the nonlinear valve. Since the closer the condition number is to unity the more separate (less collinear) are the columns of the matrix [78], the much smaller condition numbers in Table 5.1 indicate the marked improvement in the quality of $\mathbf{\Phi}_\gamma$ when scaled by $\delta\Psi_i$. Given the much improved solutions obtained by scaling, the results shown henceforth are obtained with scaled $\mathbf{\Phi}_\gamma$.

Table 5.1. Range of condition numbers of the structural sensitivity matrix $\mathbf{\Phi}_\gamma$ and the lowest error found during control restructuring with and without scaling of $\mathbf{\Phi}_\gamma$ by $\delta\Psi_i$ from Eq. (2.14)

Reference	Condition Number of Φ_γ		Lowest Error	
Value	unscaled	scaled	unscaled	scaled
1	1.61 - 12.16	2.02 - 2.07	2.61	1.35
2	1.69 - 6.95	1.80 - 2.68	4.37	2.50
3	2.13 - 4.94	1.07 - 4.69	6.10	2.65
4	10.03 - 14.05	1.09 - 2.67	8.18	3.99
5	13.37 - 13.53	1.09 - 4.48	11.38	6.10

5.5.2 Case-specificity of the Restructured Controllers

Another, and not necessarily a desirable, feature of the restructured controllers is their case-specificity, which is rooted in the search mechanism for the exponents γ_i in Eq. (2.8). As in any gradient-based search, the robustness of the solution depends on the convexity of the error surface presented in the training scenario and its form is at the mercy of the search mechanism (in this case NLS). As such, the choice of the desired response y^d plays a central role in the formulation of the solution. We have observed, for instance, that the more distant the target is from the initial closed-loop response, the better chance there is of finding a radically dissimilar controller. To illustrate the case-specificity of restructured

controllers, consider the controllers obtained at different reference values for the nonlinear valve in Table 5.2. Although the form of the restructured controllers are similar for reference values 1, 2, and 4, and for 3 and 5, they are not uniform across all reference values. To bypass this case-specificity aspect of controller restructuring, one can simultaneously reconstruct the controller for several reference values by using a step-wise reference, as is demonstrated for the nonlinear valve below.

Table 5.2. Restructured controllers obtained at different reference values for the nonlinear valve

Reference Value	Restructured Controller
1	$K_p\epsilon\left(\left\|\int \epsilon \mathrm{d}t\right\|\right)^{0.27} + K_i sgn\left(\int \epsilon \mathrm{d}t\right)\left(\left\|\int \epsilon \mathrm{d}t\right\|\right)^{0.80}$
2	$K_p\epsilon\left(\left\|\int \epsilon \mathrm{d}t\right\|\right)^{0.19} + K_i sgn\left(\int \epsilon \mathrm{d}t\right)\left(\left\|\int \epsilon \mathrm{d}t\right\|\right)^{0.82}$
3	$K_p sgn(\epsilon)\left\|\epsilon\right\|^{1.15} + K_i sgn\left(\int \epsilon \mathrm{d}t\right)\left(\left\|\int \epsilon \mathrm{d}t\right\|\right)^{0.81}$
4	$K_p\epsilon\left(\left\|\int \epsilon \mathrm{d}t\right\|\right)^{0.15} + K_i sgn\left(\int \epsilon \mathrm{d}t\right)\left(\left\|\int \epsilon \mathrm{d}t\right\|\right)^{0.78}$
5	$K_p sgn(\epsilon)\left\|\epsilon\right\|^{1.08} + K_i sgn\left(\int \epsilon \mathrm{d}t\right)\left(\left\|\int \epsilon \mathrm{d}t\right\|\right)^{0.78}$

5.6 Restructured Controllers

The controllers used for the nonlinear valve and inverted pendulum were restructured by MSAM according to the configuration in Fig. 5.1. The desired response y^d used for the nonlinear valve was that of the customized solution and the one for the inverted pendulum was that of the linear controller to the lowest magnitude ($\delta = 15$) impulse applied to the pendulum. The coupled $\hat{f}_i$ in Eq. (2.8) were the absolute values of the other variables to avoid imaginary numbers due to exponentiation of negative numbers. The restructured controllers obtained for the above platforms are discussed separately.

5.6.1 Controller for the Nonlinear Valve

As was discussed earlier, in order to obtain a more generalized controller across the reference value range, restructuring for the nonlinear valve was performed with a staircase reference profile that included three reference values, as shown in Fig. 5.7. Restructuring

was performed using the closed-loop response of the customized controller as the desired response ("desired" in Fig. 5.7), adapting each candidate controller for 15 iterations in the round robin phase and the best controller for 50 iterations in the final phase. The candidate models in the round robin phase were generated from the PI controller: $K_p\epsilon(t) + K_i \int \epsilon(t)\mathrm{d}t$ using $[f_1, f_2] = \left[|\epsilon|, |\int \epsilon \mathrm{d}t|\right]$ in Eq. (2.8). Controller restructuring resulted in

$$u(t) = K_p\epsilon + K_i\left(\int \epsilon \mathrm{d}t\right) \quad \Longrightarrow$$
$$u(t) = K_p\epsilon(t)\left(\left|\int \epsilon(t)\mathrm{d}t\right|\right)^{0.02} + K_i sgn\left(\int \epsilon(t)\mathrm{d}t\right)\left(\left|\int \epsilon(t)\mathrm{d}t\right|\right)^{0.94} \tag{5.7}$$

The response of this controller is named "restructured" in Fig. 5.7. Next we used the NLS to adapt the parameters of the controller from the initial values of $K_p = 0.1$ and $K_i = 0.1$ to obtain a closer fit to the target. The tuned parameter values were $K_p = 0.0941$ and $K_i = 0.0543$, producing the improved response "tuned restructured" in Fig. 5.7.

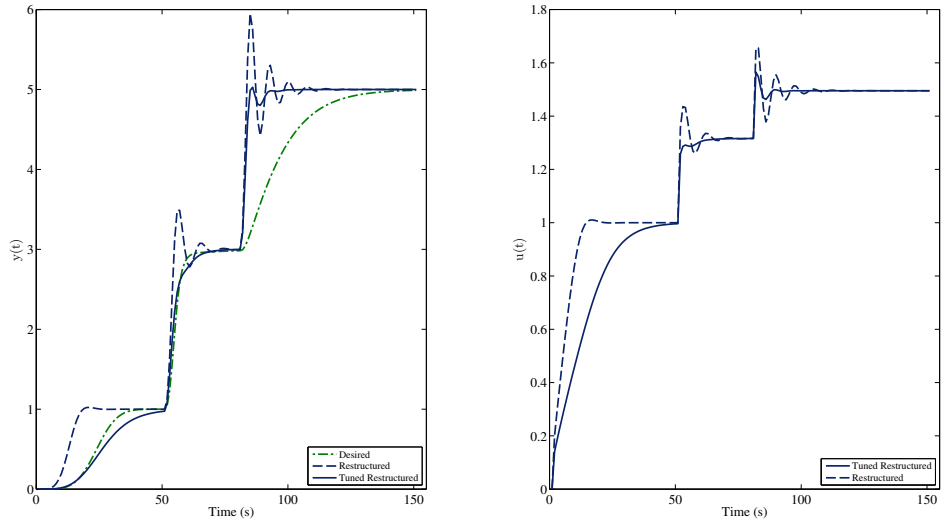

Figure 5.7. Step response and control effort of the restructured controller before and after tuning its parameters by NLS shown with the desired response used for control restructuring

As a benchmark, the response of the final restructured controller ("tuned restructured") is compared with the customized response in Fig. 5.8. The results clearly indicate a more consistent rise time of the restructured controller than the customized controller (see Fig. 5.2).

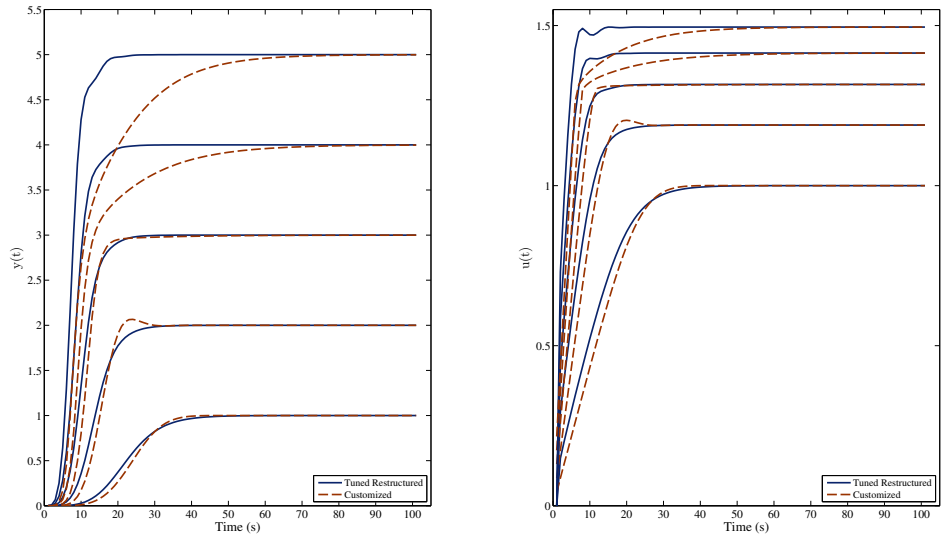

Figure 5.8. Step response of the restructured controller and its control effort compared with those of the customized controller (Fig. 5.2) at different reference values

5.6.2 Controller for the Inverted Pendulum

For the inverted pendulum, the candidate controllers were generated from the state feedback controller: $K_1x + K_2\dot{x} + K_3\theta + K_4\dot{\theta}$ using $[f_1, f_2, f_3, f_4]$ in Eq. (2.8) as $\left[|x|, |\dot{x}|, |\theta|, |\dot{\theta}|\right]$. To invoke the nonlinearity of the pendulum, an impulse magnitude of $\delta = 18$ (see Fig. 5.6) was applied to the cart with the closed-loop response of the linear controller to an impulse magnitude of $\delta = 15$ was used as the desired response. Each candidate controller was adapted for 15 iterations in the round robin phase and the best controller was adapted for 50 iterations in the final phase. The restructured controller had the form

$$u(t) = -K_1x(t) - K_2\dot{x}(t) - K_3\theta(t) - K_4\dot{\theta}(t) \quad \Longrightarrow \quad u(t) = -K_1x(t)\left|\dot{\theta}(t)\right|^{0.04} - K_2\dot{x}(t)\left|\dot{\theta}(t)\right|^{0.02} - K_3 sgn(\theta(t))\,|\theta(t)|^{0.92} - K_4 sgn(\dot{\theta}(t))\left|\dot{\theta}(t)\right|^{1.03} \tag{5.8}$$

The responses and control efforts of the restructured and linear controllers to an impulse magnitude of $\delta = 18$ are shown in Fig. 5.9 along with the desired response.

As benchmark, the impulse responses of the restructured controller is compared with that of the linear controller in Fig. 5.10 for different impulse magnitudes. Both the responses and

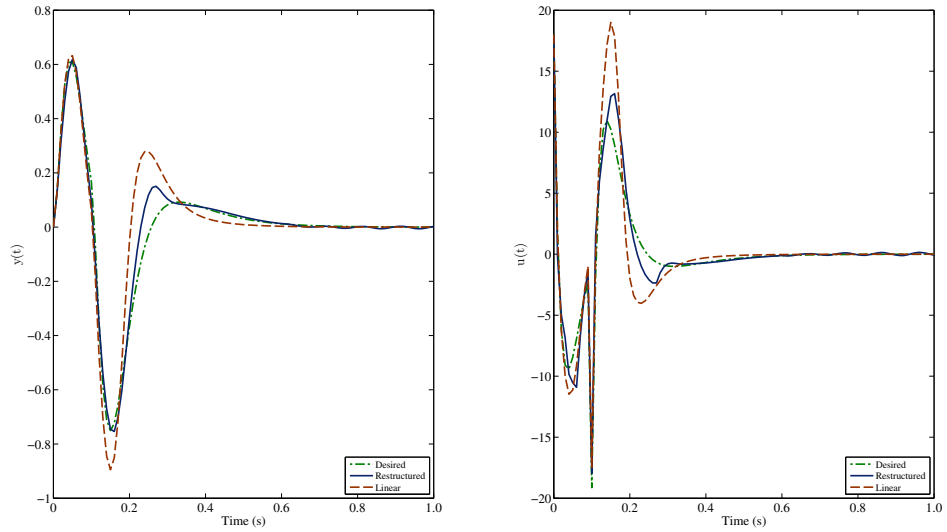

Figure 5.9. Impulse response and control effort of the restructured controller and those of the linear controller shown with the desired response used for control restructuring

control efforts of the restructured controller are significantly more consistent than those of the linear controller for different impulse magnitudes. This consistency is due in part to the quicker response of the restructured controller to state perturbations, providing the capacity to cope with impulses of higher magnitude, as discussed in the next section.

5.7 Analysis

Even though the results are anecdotal, they can be used to analyze several aspects of the restructured controllers by MSAM. One such aspect is the response of the restructured controllers to conditions absent in training, such as measurement noise, disturbances, and reference magnitudes beyond those used for training. A second aspect is the sensitivity of the restructured controllers to training conditions. A third aspect is the form and behavior of restructured components of the controller in comparison to their initial counterparts. Yet a fourth aspect is the receptiveness of the restructured controllers to parameter tuning.

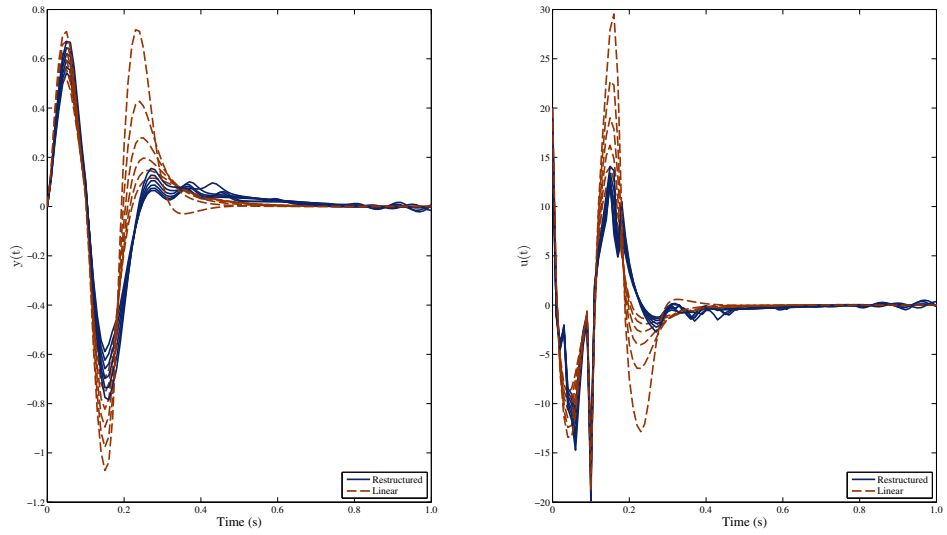

Figure 5.10. Impulse response and control effort of the restructured controller and those of the linear controller at impulse magnitudes of 15-20

5.7.1 Unrepresented Conditions

One aspect of controller performance is its response to additive measurement noise. Another aspect is its disturbance rejection capacity. A third aspect is its response to reference magnitudes not included in training.

To evaluate the performance of restructured controllers in presence of noise, band-limited noise at the signal-to-noise ratio of 18 (at $r = 1$) to 33 (at $r = 5$) was added to the output of the plant with nonlinear valve. Controller responses were tested ten times for different noise values, as shown in Fig. 5.11. The results indicate that the closed-loop responses with the restructured and customized controllers are similarly affected by measurement noise with a smaller variation observed in the control efforts.

The disturbance rejection capacity of the controllers were evaluated with unit step disturbances applied both before and after the plant $G_0(s)$ in Fig. 5.2. The closed-loop responses of both the restructured and customized controllers are shown in Fig. 5.12. The results indicate much more agile disturbance rejection by the restructured controller at higher reference values, replicating the faster step response of these controllers at higher reference values.

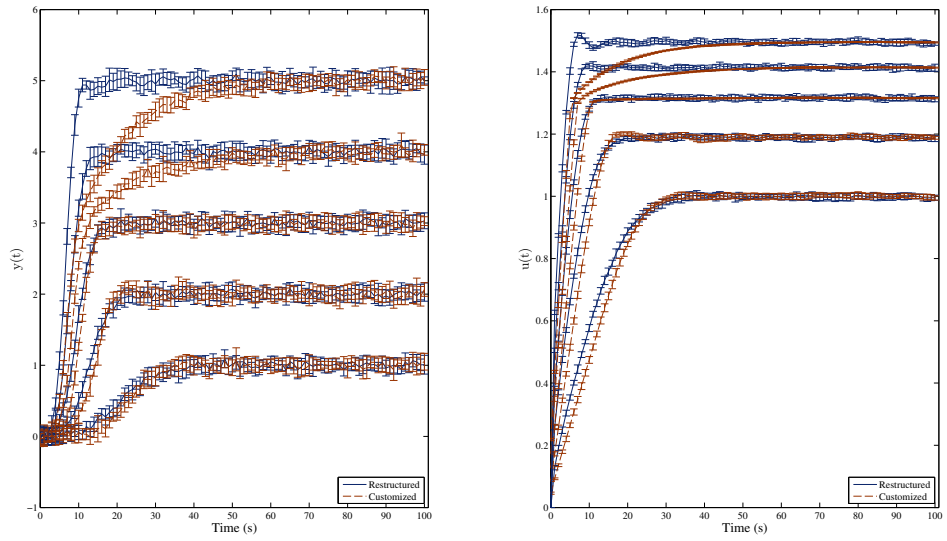

Figure 5.11. Closed-loop step response and control effort range of the restructured and customized controllers in presence of additive band-limited measurement noise at the approximate signal-to-noise ratio of 18 at $r = 1$ to 33 at $r = 5$

To evaluate the controllers regulation capacity for levels not encountered in training, the closed-loop step responses of the restructured controller are compared to the step responses of the nonlinear valve with the restructured and customized controllers were for step sizes of 6-15, as shown in Fig. 5.13. Similarly, the closed-loop impulse responses of the inverted pendulum with the restructured and linear controllers were obtained at impulse magnitudes of 21-33. The linear controller was found to be deficient in maintaining upward position for the pendulum for impulse magnitudes of 27 and higher. The responses obtained with the restructured controller for impulse magnitudes of 27-33 are shown in Fig. 5.14. The results in Fig 5.13 indicate that the restructured controller starts having oscillatory behavior at step sizes 9 and higher, while the customized solution provides continually more sluggish response at these higher steps. The results in Fig. 5.14, however, are more complimentary of the restructured controller, as they show responses that are unattainable by linear control.

Of interest is also the robustness of the restructured controllers to modeling inaccuracies. To evaluate their robustness, the closed-loop responses for the valve were generated with the valve nonlinearities of $f(u) = u^{3.5}$ and $f(u) = u^{4.5}$, and those of the inverted pendulum

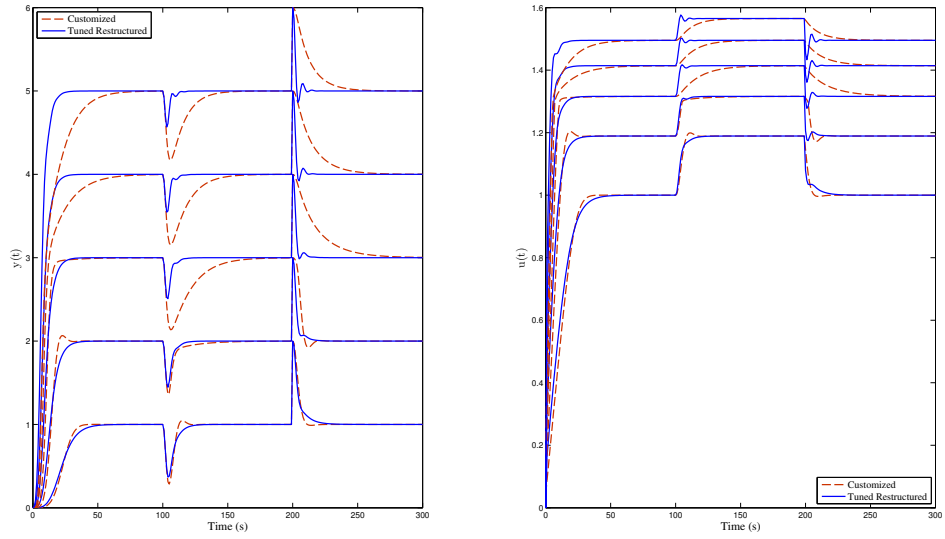

Figure 5.12. Closed-loop responses and control efforts of the nonlinear valve with parameter-tuned restructured and customized controllers to unit step disturbances before $G_0(s)$ in Fig. 5.2 (at time 100) and after $G_0(s)$ (at time 200)

were obtained for pendulums with 10%, 20%, and 30% smaller mass. The responses for the nonlinear valve are shown in Fig. 5.4 and those for the inverted pendulum are in Fig. 5.16. The responses for the restructured controller in Fig. 5.15 are quite similar in contrast to those of the customized controller for different valve models even though the controller was restructured for the nominal valve model of $f(u) = u^{4.0}$, indicating the considerable robustness of the restructured controller to unknown valve nonlinearity. Similarly, the responses of the restructured controller in Fig. 5.16 are very close for different pendulum masses relative to those of the linear controller, indicating the lower than the linear controller's sensitivity of the restructured controller to the modeled inaccuracy in pendulum mass.

5.7.2 Sensitivity to Training

As was discussed earlier and depicted by the controller forms in Table 5.2, the training conditions influence the controller forms. This sensitivity to training conditions prompted expanding the diversity of reference values by using a stair case format for restructuring the controllers for the nonlinear valve. It, therefore, behooves us to examine the sensitivity of the controller forms to different stair case scenarios. Similarly, the restructured controller for

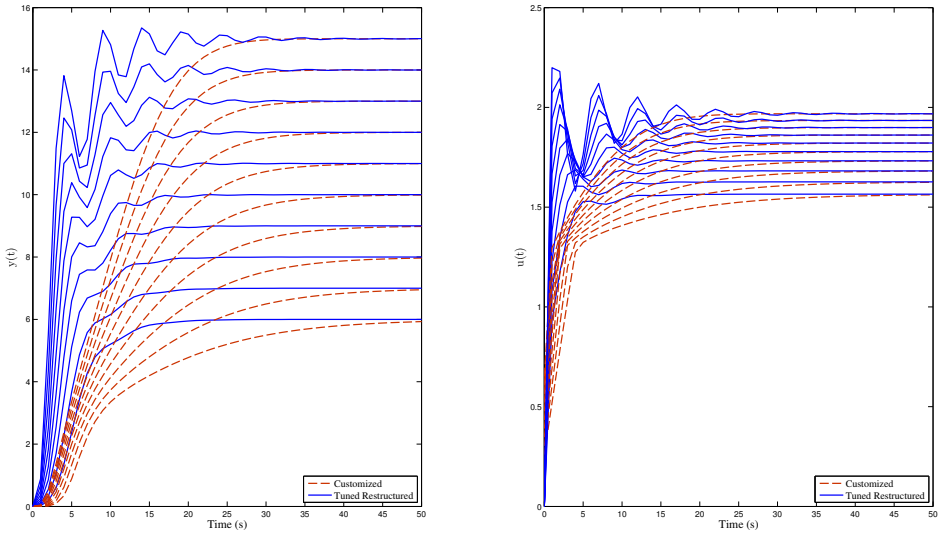

Figure 5.13. Closed-loop responses and control efforts of the nonlinear valve with parameter-tuned restructured and customized controllers at higher step sizes (6 - 15) than those (1 - 5) used for restructuring.

the inverted pendulum was sought using different impulse magnitudes. It is also pertinent to examine the diversity of controller forms obtained at different impulse magnitudes. To this end, the controller forms obtained for the nonlinear valve and inverted pendulum in different training cases are shown in Table 5.3. The results indicate two controller forms found across the ten different stair case combinations (e.g., 1,2,3; 1,3,5; 2,3,4; etc.) for the nonlinear valve and three controller forms for the inverted pendulum at three different impulse magnitudes. The difference between the controller forms for the nonlinear valve is in the first component wherein the ϵ is coupled with itself, in the first case, and with its integral, in the second case. The restructured controller forms for the inverted pendulum, however, are quite diverse and can be compared better through their simulated behavior, as presented below in the following subsection.

5.7.3 Controller Components

The different forms obtained for the restructured controllers raise two important questions: (1) how different are the individual components of the controller from each other in different forms and from their counterparts in the initial controller, and (2) how differently

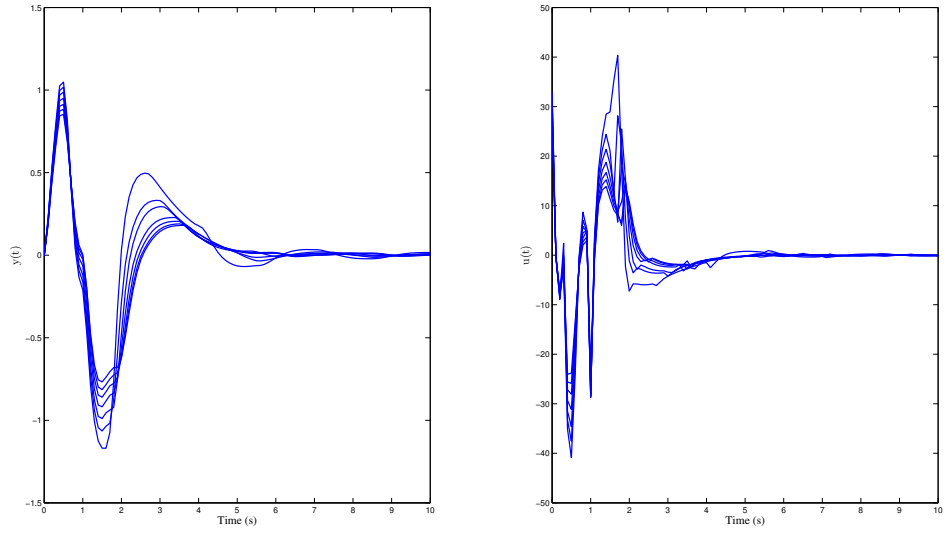

Figure 5.14. Closed-loop impulse responses and control efforts of the inverted pendulum with restructured controller (obtained with the impulse magnitude of 20) at impulse magnitudes of 27-33 that are beyond the capacity of the linear controller

do they contribute to the total control effort of each controller. To address these questions, the numerical values of the individual components (see Table 5.3) at different sample points were obtained during simulation, as shown in Fig. 5.17 for the nonlinear valve and in Fig. 5.18 for the inverted pendulum. The results in Fig. 5.17 indicate that the proportional effect "$K_p sgn\left(\epsilon(t)\right)\left|\epsilon(t)\right|^{(\gamma_1+1)}$" provides a smaller portion of the overall effort than "$K_p\epsilon(t)\left|\int \epsilon \mathrm{d}t)\right|^{\gamma_1}$", and that it has a nonzero value initially because of its entire dependence on the "$\epsilon(t)$", whereas its counterpart is initially null due to its dependence on "$\int \epsilon \mathrm{d}t$" before it rises rapidly to its maximum value. The integral components, which have the same form, only differ slightly because of the differences in the magnitude of "$\int \epsilon \mathrm{d}t$" in the two simulation runs.

The results in Fig. 5.18 show a much more nuanced difference of the controller components, since they not only differ in form but also the coefficient and exponent values. For instance, consider the similar in form "x effort" of the restructured controller at $\delta = 18$ and $\delta = 20$. As shown in the first row of Fig. 5.18, their behavior is much more different from those at $\delta = 18$ and $\delta = 19$ that are different in form. We attribute this difference to the confluence of the

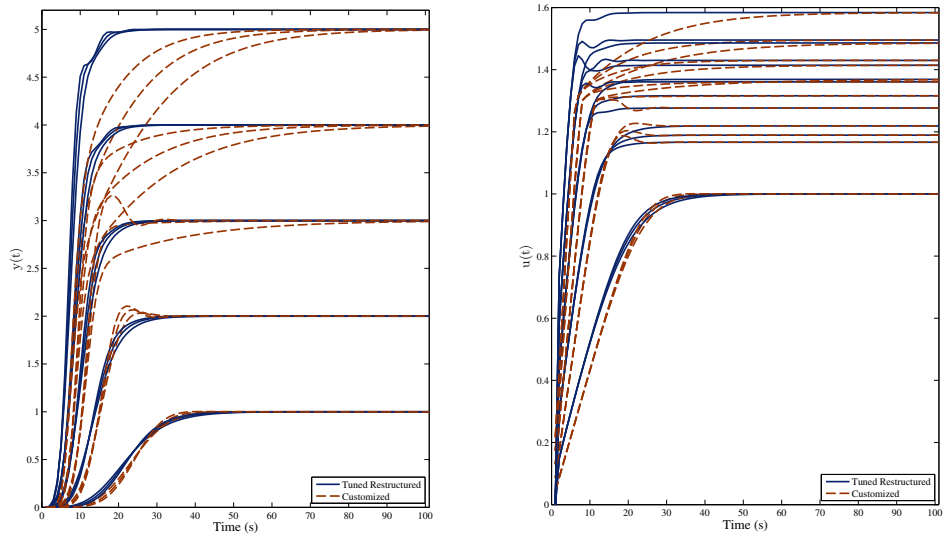

Figure 5.15. Step response and control effort of the restructured controller compared with those of the customized controller (Fig. 5.2) affected by inaccurate valve nonlinearities

other components. But perhaps the most interesting observation of the results in Fig. 5.18 is the similarity between the total control efforts shown in the last row of this figure, despite the very different behavior of individual components.

5.7.4 Receptiveness to Parameter Tuning

Another aspect of the restructured controllers is their potential improvement by parameter tuning [62, 49]. To examine their sensitivity to parameter tuning, shown in Table 5.4 are the mean percent of reduced error achieved by NLS-based parameter tuning for the linear and restructured controllers. As indicated by the results in Table 5.4, the restructured controller for the nonlinear valve benefits far more than the initial PI controller from parameter tuning, whereas the benefits of parameter tuning to the restructured controller for the inverted pendulum are slightly less than those of its starting linear controller.

5.8 Discussion

- *Stability:* As with any controller design, of concern is the stability of the closed-loop systems containing the restructured controllers. Fortunately, a fundamental benefit of

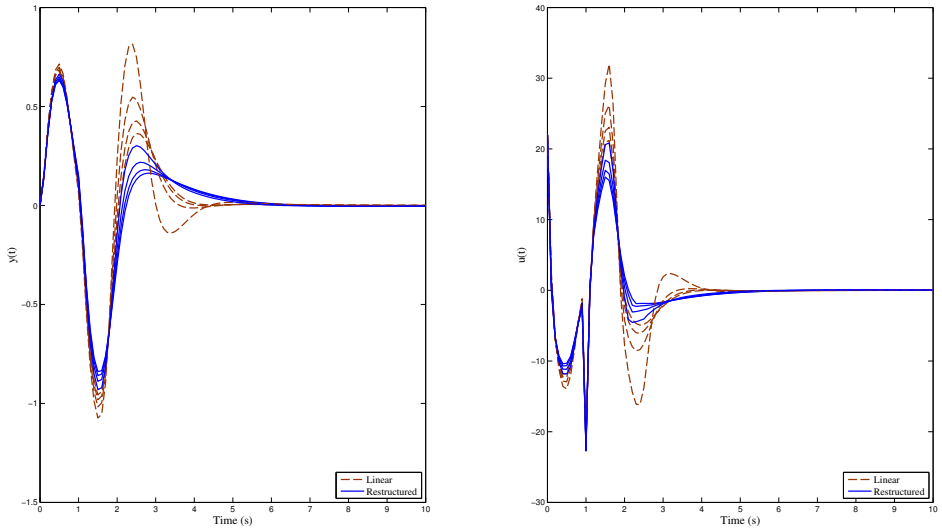

Figure 5.16. Closed-loop impulse responses and control efforts of the inverted pendulum with the restructured and linear controllers when inaccuracies of 0%, 10%, 20% and 30% exist in the pendulum mass

the proposed restructuring format, in Fig. 5.1, is its empirical evaluation of the candidate controllers in simulation. Since MSAM is designed to produce a controller that is at least better than the starting controller, it discards any candidate controller that is inferior in performance to other candidate controllers as well as to the initial controller. Given that the instability of the system is a natural criterion in this performance evaluation, the solutions delivered by MSAM are guaranteed to be closed-loop stable within the bounds of the simulation.

- *Reachability:* In general, MSAM is additive by nature, designed to adapt a potentially inadequate initial controller by adding coupling to its individual components. Accordingly, this method is suited to restructuring initial controllers that are simple in form, as they are guaranteed to be less complex than their restructured version. Furthermore, MSAM operates with the assumption that a potentially superior restructured controller is reachable by prescribed adjustments to the components of the starting controller. To this end, the selection of the adjustments $\hat{f}_i$ in Eq. (2.8) is of paramount importance.

Table 5.3. Restructured controllers obtained at different stair cases for the nonlinear valve and impulse magnitudes for the inverted pendulum

	Restructured Controller
Step Sizes	Nonlinear Valve
1,2,3 and 2,3,5	$K_p sgn\left(\epsilon(t)\right)\left\|\epsilon(t)\right\|^{(\gamma_1+1)} + K_i sgn(\int \epsilon \mathrm{d}t)\left\|\int \epsilon \mathrm{d}t\right\|^{(\gamma_2+1)}$
all others	$K_p \epsilon(t)\left\|\int \epsilon \mathrm{d}t)\right\|^{\gamma_1} + K_i sgn(\int \epsilon \mathrm{d}t)\left\|(\int \epsilon \mathrm{d}t)\right\|^{(\gamma_2+1)}$
Impulse Magnitude	Inverted Pendulum
$\delta = 18$	$K_1 x(t)\left\|\dot{\theta}(t)\right\|^{\gamma_1} + K_2 \dot{x}(t)\left\|\dot{\theta}(t)\right\|^{\gamma_2} + K_3 sgn\theta(t)\|\theta(t)\|^{\gamma_3+1}$ $+K_4 sgn(\dot{\theta}(t))\left\|\dot{\theta}(t)\right\|^{\gamma_4+1}$
$\delta = 19$	$K_1 x(t)\left\|\theta(t)\right\|^{\gamma_1} + K_2 sgn(\dot{x}(t))\left\|\dot{x}(t)\right\|^{(\gamma_2+1)} + K_3 sgn(\theta(t))\left\|\theta(t)\right\|^{(\gamma_3+1)}$ $+K_4 \dot{\theta}(t)\left\|\dot{x}(t)\right\|^{\gamma_4}$
$\delta = 20$	$K_1 x(t)\left\|\dot{\theta}(t)\right\|^{\gamma_1} + K_2 \dot{x}(t)\left\|\theta(t)\right\|^{\gamma_2} + K_3 \theta(t)\left\|\dot{x}(t)\right\|^{\gamma_3}$ $+K_4 \dot{\theta}(t)\left\|\theta(t)\right\|^{\gamma_4}$

- *Scalability:* The scalability of MSAM depends on the number of candidate controllers considered during the round robin stage. Given that with n adjustments applied to Q components, Q^n candidate models need to be examined during the round robin phase, the selection process can become overwhelming if the controllers are examined sequentially. Fortunately, the examination of individual candidate controllers is independent of the others, therefore, this phase can be run in parallel, reducing the computation time to Q^n/p, with p number of processors. For large-scale problems that cannot

Table 5.4. Percent reduction by parameter tuning in the absolute sum of the error between the closed-loop response and its target for both the linear and restructured controllers to assess their potential for improved performance by parameter tuning

	Mean Percent Reduction in Error	
	Linear Controller	Restructured Controller
Nonlinear Valve	11.86	32.04
Inverted Pendulum	84.50	61.56

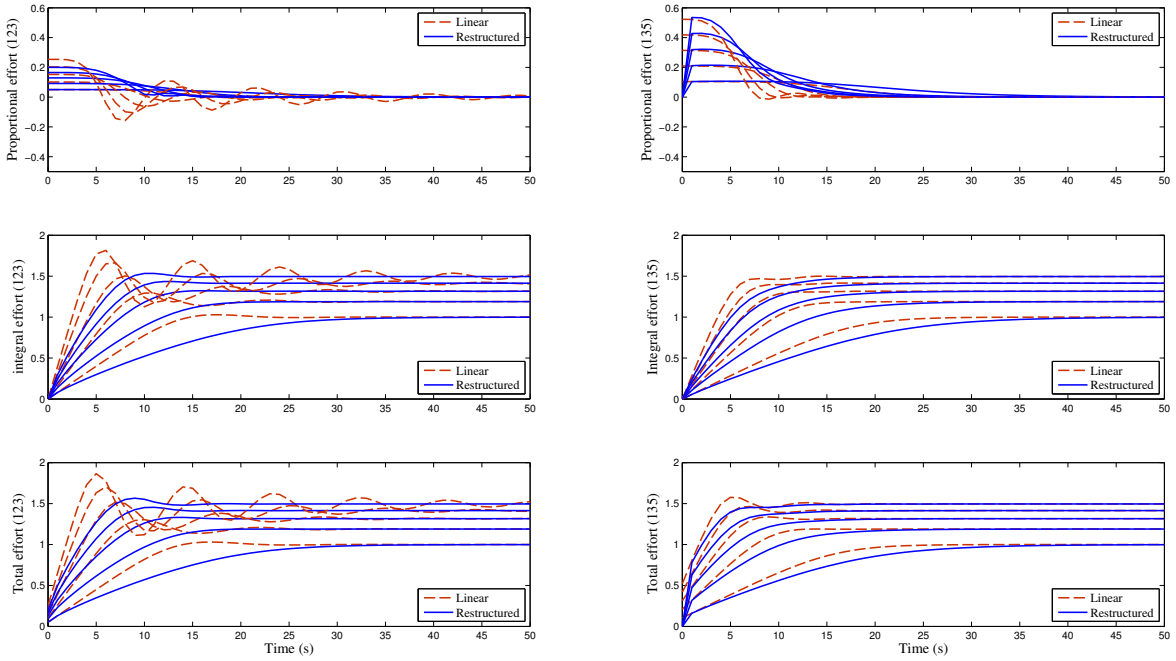

Figure 5.17. Components of the control effort by the linear and restructured controllers of the two forms in Table 5.3 in response to steps of magnitudes 1-5

be exhaustively searched, one can choose a subset of round robin controllers that are mechanistically plausible.

- *Algorithmic issues:* As with any other gradient-based search routine, the search process may be sensitive to several parameters. One such parameter is the size of the perturbation $\delta\gamma_i$ in Eq. (2.14) used for computing the structural sensitivities. Another parameter is the initial value of μ in Eq. (2.17) that is adjusted at each iteration step. A third parameter is the perturbation size of the individual parameters used for computing $\partial\hat{y}/\partial\mathbf{\Theta}$ in Eq. (2.14). Yet a fourth parameter is the fitness function used to evaluate the candidate models, currently formulated to consider the size of the error as well as the correlation of the candidate output with its target. The sensitivity of the search process to these parameter will depend upon the convexity of the error surface, and needs to be evaluated in the context of each problem.

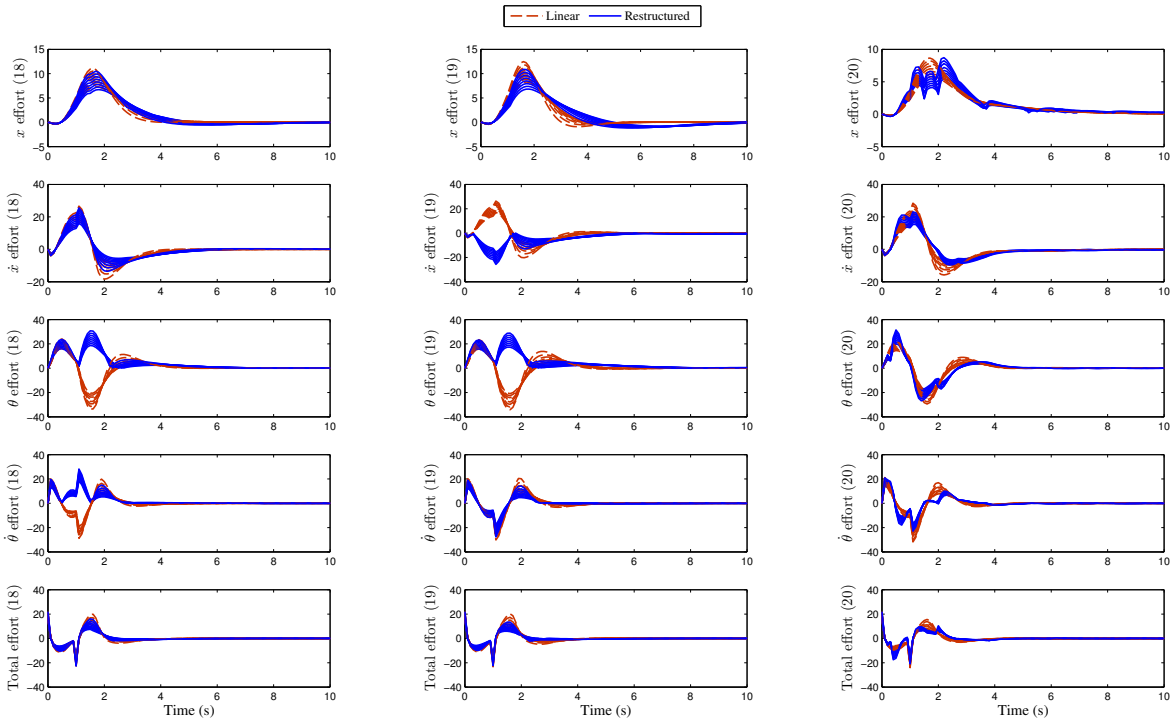

Figure 5.18. Components of the control effort by the linear and restructured controllers for the inverted pendulum in response to impulse magnitudes of 15-22

5.9 Conclusion

A method of gradient-based search is introduced for adapting the structure of controllers. The proposed method is demonstrated in application to two benchmark problems and its solutions are analyzed in response to conditions not introduced in training. The case study results indicate that this method is effective in upgrading the structure of the linear controllers designed according to linearized models of the plants to nonlinear controllers that can better cope with the nonlinearities of the plants.

5.10 Acknowledgments

I would like to thank Kourosh Danai and Kushal Sahare for their work in contributing many of the results and analyses in this chapter.

CHAPTER 6

AUTOMATIC IDENTIFICATION OF CLOSED-LOOP WIND TURBINE DYNAMICS VIA EVOLUTIONARY MULTIOBJECTIVE OPTIMIZATION

6.1 Summary

Modern industrial-scale wind turbines are nonlinear systems that operate in turbulent environments[1]. As such, it is difficult to characterize their behavior accurately across a wide range of operating conditions using physically meaningful models. Customarily, the models derived from wind turbine data are in 'black box' format, lacking in both conciseness and intelligibility. To address these deficiencies, we use a recently developed symbolic regression method to identify models of a modern horizontal-axis wind turbine in symbolic form. The method uses evolutionary multiobjective optimization to produce succinct dynamic models from operational data while making minimal assumptions about the physical properties of the system. We compare the models produced by this method to models derived by other methods according to their estimation capacity and evaluate the trade-off between model intelligibility and accuracy. Several succinct models are found that predict wind turbine behavior as well as or better than more complex alternatives derived by other methods. We interpret the new models to show that they often contain intelligible estimates of real process physics.

[1]The work in this chapter was presented at the 2015 ASME Dynamic Systems and Controls Conference [106] and is the basis of a journal publication [105].

6.2 Introduction

As wind energy grows across the globe and new offshore wind turbine installations encounter new operating environments, the models that inform the design and control of these multimillion-dollar machines become increasingly important. Typical multimegawatt wind turbines exhibit nonlinear behavior and are subject to wind (and sometimes wave) disturbances that are often hard to estimate. These properties make the simulation of their dynamics not only challenging but also site-dependent, because of the influence of wind, wave, and foundation characteristics. Accordingly, the first-principles models of wind turbines, such as the one embedded in the aero-hydro-elastic simulation tool FAST [81], are prone to cumulative discrepancies between prediction and reality. These models are also computationally expensive to run because of their fairly comprehensive representation of wind turbine dynamics. Although the use of engineering models is fundamental to the structural design and loads analysis process, model-based controllers preferably rely on a customized model of the real system in the field, rather than a first-principles model that may miss key elements present in the real system[74].

As an alternative to potentially inaccurate and computationally expensive first-principles models, empirical models of wind turbines are obtained from experimental data to provide a customized representation of the wind turbine. These models are usually in the form of autoregressive moving-average (ARMAX) models [74, 209, 75, 208], neural networks [101], or fuzzy logic models [17], among others, to provide the structural flexibility for adapting the model according to the measured observations. Although these empirical models provide an effective means of estimation/prediction, they have the major drawback of lacking transparency about the physics of the process [12]. This lack of transparency obscures the knowledge of the process that is gained through their development. Ideally, the model should not only be accurate, but intelligible so that the user acquires the insight attained through the model's development. A well-formed model serves two purposes: (i) it improves knowledge of the underlying dynamics of the system; and (ii) it improves the ability of the wind turbine controller to extract power and minimize loads on the turbine.

In order to achieve these goals, we evaluate the applicability of the proposed ELGP method in identifying wind turbine models based on experimental data collected in normal closed-loop operation from the three-bladed Controls and Advanced Research Turbine (CART3), a turbine maintained by the National Renewable Energy Laboratory (NREL). The paper is organized as follows. First, we present a brief overview of wind turbine mechanics. We then review previous system identification work. Next, the problem formulation as sought by multiobjective optimization is presented, followed by a description of the proposed ELGP method. We then detail the wind turbine identification procedure and analyze results pertaining to local and global models of the wind turbine. The paper concludes with a discussion of the intelligibility of the identified models as they inform the physics of the process.

6.3 Wind Turbine Mechanics

Identification of wind turbine models is a difficult undertaking because of the many layers of nonlinearity governing their behavior. Moreover, modern horizontal-axis wind turbines (HAWTs) are controlled using variable-speed and variable-blade pitch operation, further complicating the dynamics. Consider for instance the steady-state aerodynamic rotor torque (Q_R) and thrust (T_R) generated by the rotor operating in freestream wind speed V, defined by:

$$Q_R = \frac{1}{2}\rho\pi R^3 C_q(\lambda,\beta)V^2 \tag{6.1}$$

$$T_R = \frac{1}{2}\rho\pi R^2 C_T(\lambda,\beta)V^2 \tag{6.2}$$

where the tip speed ratio $\lambda = \Omega R/V$ relates the rotor speed Ω to the wind speed V, ρ is the air density, R is the rotor radius, β is the pitch angle of the blades (assumed pitching collectively). C_q and C_T are the torque and thrust coefficients, respectively, defining the corresponding generated lift as functions of λ and β. The overall C_q is a function of local

aerofoil drag and lift coefficients C_d and C_l, local incidence angle with the wind, ϕ, and local tip speed ratio λ_r, defined by the strip theory calculation of C_q, as:

$$C_q = (8/\lambda^3) \int_{\lambda_h}^{\lambda} \sin^2\phi(\cos\phi - \lambda_r \sin\phi)(\sin\phi + \lambda_r \cos\phi)[1 - \frac{C_d}{C_l}\cot\phi]\lambda_r^2 d\lambda_r \tag{6.3}$$

Because it is difficult to obtain the lift and drag coefficients at each position along the blade due to small inconsistencies in fabrication and local shape deflections, they are often estimated empirically [129]. The inaccuracy of estimated nonlinear coefficient surfaces C_q and C_T, compounded with the measurement uncertainty and stochasticity of V, impedes prediction of the aerodynamic torque and thrust response of the system.

Control actions are limited to actuating the collective pitch β, the generator torque T_G, and the yaw angle ψ. Because of the highly nonlinear nature of the wind turbine behavior, a pitch action of the same magnitude may result in very different aerodynamic forces depending on the instantaneous wind speed and rotor speed, requiring the employment of gain scheduling for pitch control [16]. In addition to aerodynamic nonlinearities, the turbine has low-frequency periodic excitations induced by the rotating blades at once-per-revolution (1P) and thrice-per-revolution (3P) that are normally within the same frequency range as the fore-aft (FA) and side-side (SS) natural frequencies of the tower, requiring the added provision of avoiding dynamic coupling between these excitations and that of the pitch control that affects Ω. Similarly, the first mode of the wind turbine drivetrain can be excited by the generator torque commands, so the generator control must account for this fundamental design objective as well. From the above anecdotes it follows that an accurate model of the wind turbine is essential for designing a reliable controller. This need for model accuracy motivates data-based modeling approaches that can account for turbine-specific observations and provide confident estimates of wind turbine behavior.

6.4 Related Work

Most system identification attempts at modeling wind turbines have focused on producing linear time-invariant (LTI) models via ARMAX models [74, 75] or modified forms of closed-

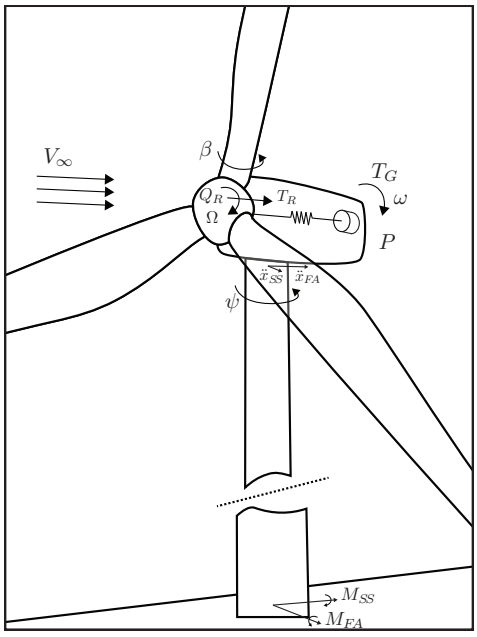

Figure 6.1. The wind turbine mechanics considered for identification.

loop subspace identification (SSID) [209, 208]. Although LTI models seem to be effective in characterizing simulated wind turbine behavior at specific operating wind speeds [74, 75], they provide only localized representation. As a remedy, SSID methods have been extended to account for the time-varying, nonlinear dynamics of the wind turbines to form global models. For example, Van der Veen [208] showed that Wiener and Hammerstein systems could be used to identify global wind turbine dynamics by providing the model with the nonlinear aerodynamic torque and thrust relations (Eqs. (6.1) and (6.2)), as well as the surface functions for (C_p) and (C_T) that vary with the tip speed ratio λ and pitch angle β. This approach, however, requires good knowledge of these two surface functions, which rely on first principles. Another approach to global modeling associates the nonlinearities with the azimuth angle of the rotor and uses a linear parameter-varying (LPV) model to conduct closed-loop identification of the wind turbine dynamics [209]. In this case, the dynamics of the turbine are assumed to vary periodically, so the matrices of the state space model are defined in terms of the azimuth position of the rotor. This approach provides good predictions of the hub moments at the rotor and tower top motion.

The above approaches, albeit in 'black-box' form, are attractive because of their incorporation of expert knowledge in modeling some of the nonlinearities and for their accommodation of control design. Ideally, however, the system identification approach has the flexibility to work when the aerodynamic properties of the wind turbine and/or the sources of its nonlinear behavior are not well-characterized. In addition, the methods above used special operating conditions in which the input actions (e.g., β, T_G) are perturbed in a pseudo-random binary fashion to minimize the correlation of output and input noise in closed-loop operation. This approach is problematic because it is not always possible for a control engineer to apply excitation signals to the operating turbine, nor is it straightforward to persistently excite the system adequately [74]. For this reason we focus our identification on normal operating data.

There have been some attempts to construct wind turbine models under similarly reduced sets of assumptions, although most focused solely on power prediction, e.g., from wind measurements [38] or from low resolution supervisory control and data acquisition (SCADA) data [102, 101]. Kusiak [102] demonstrated that a neural network model and a controller designed via evolutionary computation could improve simulated power output in below-rated conditions. A drawback of this approach is that the models need to be periodically regenerated to continue to perform well, suggesting an overfitting scenario. The method we propose differs from typical data mining in that it precludes structural assumptions for the model and focuses on the derivation of simple, explicative models that are valuable for their intelligibility in addition to their estimation capacity.

6.5 Problem Statement

Recall from §1.1 that in the search for the correct model form M^*, GP typically attempts to solve the problem:

$$\text{minimize } f(M) \quad \text{subject to } M \in \mathfrak{S} \tag{6.4}$$

where $\mathfrak{S}$ is the space of possible models M, and f denotes a minimized fitness function. Typically, a single fitness function quantifies the difference between the target output y and a

candidate output $\hat{y}$; however, there is often more than one objective to consider for evaluating the model, in which case the problem becomes:

$$\begin{aligned} &\text{minimize } f_j(M), \qquad j = 1, \ldots, J \\ &\text{subject to } M \in \mathfrak{S} \end{aligned} \tag{6.5}$$

The multiple objective function $\mathbf{f} = [f_1 \quad ... \quad f_J]$ would ideally yield a set of nondominated solutions $\widetilde{\mathbf{M}} = \{\widetilde{M}_1 \quad ... \quad \widetilde{M}_n\}$, comprising the set of solutions that are Pareto-optimal in $\mathfrak{S}$, where model dominance is defined as:

Definition: Model M_1 dominates M_2; i.e., $(M_1 \prec M_2)$ if $f_j(M_1) \leq f_j(M_2)\ \forall j$ and $f_j(M_1) < f_j(M_2)$ for at least one j.

In lieu of an exhaustive search of $\mathfrak{S}$, the goal of EMO is to return a set of models $\widehat{\mathbf{M}}$ as close to the Pareto-optimal set $\widetilde{\mathbf{M}}$ as possible. It may be easier to represent an arbitrary set of data by a complex model, but it is more difficult to understand and generalize the information content of such a model. Therefore, the solutions from the search must provide a balanced trade-off between accuracy and complexity. Population-based optimization methods like GP are well-suited to address the conflicting objectives of accuracy and conciseness because the solution set $\widehat{\mathbf{M}}$ offers multiple candidate models for approximating $\widetilde{\mathbf{M}}$. In the following section we describe a recent symbolic regression method designed to address such a trade-off that is used to conduct the identification of the wind turbine models in this paper.

6.6 Proposed Method

We use the ELGP algorithm described in Ch. 3 as the identification method for this task. To put the focus more strongly on intelligibility for this application, complexity is explicitly used as an objective for identifying models. The follow section describes the objectives used to guide search.

6.6.1 Evolutionary Multiobjective Optimization

We use three objectives to drive evolutionary pressure during optimization: variance accounted for (VAF), model complexity, and the age of the program in the population. The first two objectives are designed to achieve model accuracy and simplicity. The third objective is used to prevent premature convergence. The three objectives are described in more detail below.

- VAF: We assess the accuracy of each candidate program $\mathbf{i}$ using the VAF metric, which characterizes the normalized variance of the prediction error as:

$$\text{VAF}\left(\mathbf{i}\right) = \max\left(0, \frac{1 - \text{var}(\text{y} - \hat{\text{y}})}{\text{var}(\text{y})}\right) \times 100 \tag{6.6}$$

  Equation (6.6) is transformed into a minimized objective function as $f_{VAF}\left(\mathbf{i}\right) = 1 - VAF\left(\mathbf{i}\right)/100$.

- Model Complexity: There are several ways to represent the complexity of a model. For example, one can count the number of nodes in the parse tree, or calculate the order of a Chebyshev polynomial fit to the model's output [211]. Here, we account for model complexity by assigning component function nonlinearities to genotype components [190]. Given the following active genotype $\mathbf{g_a} = \left[\begin{array}{ccc} g_{a_1} & \ldots & g_{a_\ell} \end{array}\right]$ for program $\mathbf{i}$, the complexity $C(\mathbf{i})$ is defined as:

$$C(\mathbf{i}) = \sum_{q=1}^{\ell} c(g_{a_q}) \tag{6.7}$$

  with the component function nonlinearities defined as:

$$c(g_a) = \begin{cases} 4 & : \quad (g_a = \log) \vee (g_a = \exp) \\ 3 & : \quad (g_a = \sin) \vee (g_a = \cos) \\ 2 & : \quad (g_a = /) \\ 1 & : \quad \text{otherwise} \end{cases} \tag{6.8}$$

- Age: Age was originally proposed as a way to layer populations during evolution [66] and later proposed as an objective in a multiobjective scheme [178]. The age of a model in the population is the number of generations since its oldest ancestor was created. To create age stratification, we introduce a new individual with age 0 to the population each generation. The use of age as an objective protects younger models from being dominated by older ones that are more fit and/or less complex. Furthermore, because younger individuals dominate older ones that may be otherwise equivalent, the introduction of age as an objective pressures the models to improve in fitness and/or complexity with increasing generations, which helps avoid premature convergence.

We implement age-fitness Pareto optimization (see §3.6.1) as the evolutionary algorithm for identification. In addition, it is important to guarantee that all solutions that are succinct and accurate are saved during optimization. With this in mind, an archive is kept updated each generation that contains all nondominated individuals according to only the metrics of VAF and model complexity. This archive provides the solutions that are explored later in this paper.

6.7 System Identification of CART3

The proposed method is evaluated in application to experimental data from the CART3 system. NREL's CART3 is instrumented with numerous sensors to make the identification of various system models possible. The nature of experimental data available from this system, including its instrumentation, data collection procedure, and control system, is described first. We then describe the settings used for ELGP, including general and problem-specific settings, followed by the types of models considered for identification. We also utilize other system identification approaches to benchmark the ELGP results.

6.7.1 CART3 System

The CART3 is a 600 kW wind turbine, down-rated to 550 kW, that acts as a test bed for field research at the National Wind Technology Center. It is a three-bladed machine that

operates with collective pitch and variable speed control. The CART3 has been instrumented extensively [45] so that data-driven models of the turbine can be established. Among the variables measured are the generator speed ω, rotor speed Ω, pitch action β, generator torque command T_G, tower top acceleration in the fore-aft ($\ddot{x}_{FA}$) and side-side ($\ddot{x}_{SS}$) directions, tower moments in the fore-aft (M_{FA}) and side-side (M_{SS}) directions, and measured power P, as shown in Fig. 6.1. In addition, an estimate of V is obtained from a meteorological tower located upwind of the turbine. Wind measurements are notoriously uncertain and although methods exist to obtain a better estimate of V [208], they assume good knowledge of C_T and rotor inertia, which are assumed unknowable in our modeling exercise.

To understand the experimental results obtained from the CART3, the controllers used by this system [16, 219] are briefly reviewed. The system consists of separate torque and pitch controllers. At wind speeds below rated-power conditions and above cut-in (Region 2 in Fig. 6.2), the blade pitch β is held constant at an estimated optimum while the generator torque T_G is adjusted proportionately to ω^2 to achieve a theoretical maximum power coefficient C_P. Conversely, at wind speeds above rated-power conditions (Region 3 in Fig. 6.2), T_G is held constant and the blade pitch β is adjusted by a proportional plus integral (PI) controller to maintain the reference generator speed at the rated power. As such, both closed-loops rely on the accuracy of the model representing the generator speed ω in terms of the corresponding control effort, as the sole input to both systems. In addition to controlled adjustments, β demand is filtered at the tower's first FA and SS modal frequencies to avoid excitation and T_G is filtered to add damping to the drivetrain's first torsional natural frequency. Although the control system is equipped to measure $\ddot{x}_{FA}$ and $\ddot{x}_{SS}$ for damping tower acceleration by adjusting β [45, 16], this control strategy was not used during these experiments.

6.7.2 Identification Procedure

For system identification, we used operating data from 14 different 5 minute operating periods. The data were collected at 400 Hz and down-sampled to 20 Hz before system identification. The data corresponded to normal operating conditions associated with wind

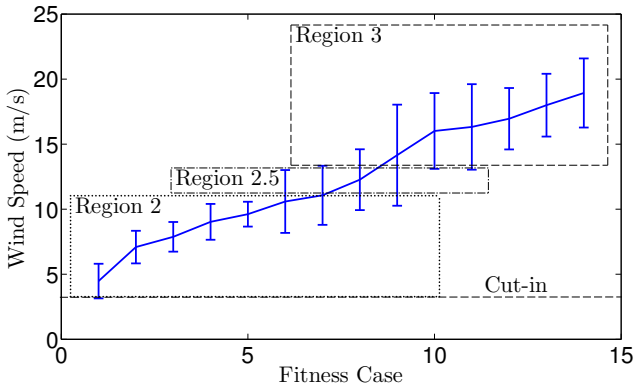

Figure 6.2. Mean wind speeds for the 5 minute data sets that comprise the training and validation sets. The bottom dotted line indicates the cut-in speed and the regions of turbine operation are marked. Bars indicate the standard deviation of the wind speed.

speeds ranging from 2.2 to 25.4 m/s, as shown in Fig. 6.2, including several start-up and shutdown events. We performed local model identification on each data set separately as well as global identification on the entire combined set (70 minutes of data). During identification, 30% of the data were chosen randomly from the set to be withheld for validation.

Models were obtained for ω, Ω, M_{FA}, M_{SS}, and P using ELGP with the settings shown in Table 6.1. We considered three types of models: static models (SMs), in the form of $M(\mathbf{u}, \boldsymbol{\Theta})$, first-order discrete-time models (DTMs), in the form of $M(\hat{y}(t_k - 1), \mathbf{u}(t_k), \mathbf{u}(t_k - 1), \boldsymbol{\Theta})$, and first-order DTMs for one-step look-ahead predictions (DTM-LAs), in the form of $M(y(t_k - 1), \mathbf{u}(t_k), \mathbf{u}(t_k - 1), \boldsymbol{\Theta})$. The main difference between DTM and DTM-LA forms is their reliance on past predictions versus experimental data. As such, DTMs evaluate the effectiveness of ELGP in a simulation-based environment, in which the output is generated entirely according to the past estimated outputs, whereas DTM-LAs evaluate the scenario of the outputs estimated according to the past values of the measured outputs. The constant values $\boldsymbol{\Theta}$ were initialized as ephemeral random constants [93] picked uniform-randomly from the range $[-10, 10]$. They were then optimized across the population each generation using a stochastic hill climbing algorithm [15]. The hill climber perturbed all constant values in the active genotype by Gaussian noise with a standard deviation equal to 10% of the value of

Table 6.1. Symbolic regression settings.

Setting	Value
Population size	2400
Crossover/Mutation	80/20%
Program length limits	[10, 100]
Ephemeral random constant range	[-10,10]
Termination criterion	2e12 (local) / 1e13 (global) gene evaluations
Function set	{+,−,∗,/,sin,cos,exp,log}

Output	Dependent variables
Ω	$\{V, \beta, T_G, t\}$
ω	$\{V, \beta, T_G, t\}$
M_{FA}	$\{V, \beta, T_G, t, \lambda, \ddot{x}_{FA}, \ddot{x}_{SS}\}$
M_{SS}	$\{V, \beta, T_G, t, \lambda, \ddot{x}_{FA}, \ddot{x}_{SS}, \psi\}$
P	$\{V, \beta, T_G, \omega\}$

the constant. These changes were kept if they resulted in a lower f_{VAF} for the individual. To prevent the search from focusing on constant optimization, and considering the insensitivity of the fitness metric f_{VAF} to linear transformations of the model output, the variables and outputs were scaled prior to adaptation, for example by rated torque, rated power, or cut-out wind speed.

A key advantage of symbolic regression is *feature selection*: the ability to select the variables to be considered in the model. We expect this feature to be instrumental in delivering parsimonious model forms that more clearly relate process inputs to model outputs as compared to the traditional system identification methods that are void of this capacity. In this regard, the models produced for global identification were compared to several model types obtained in the form of multiple regression, 20^{th}-order auto-regressive exogenous (ARX) models, and nonlinear ARX neural networks (NARX-NN). The NARX-NN contained 10 hidden layers and the weights were trained using back-propagation with the Levenberg-Marquardt algorithm.

6.8 Results

The performance of the identified local models by ELGP is summarized in Table 6.2. The results correspond to the final model with the maximum VAF from training. The results

indicate that the accuracy of the models of Ω, ω, and P is excellent in all cases except for $\bar{V} = 18.0$ m/s, where the response of the system was flat (i.e., $var(\Omega) <$ 5e-4), obscuring the dynamics. The DTM model has a better performance for this case. The other two models, M_{FA} and M_{SS}, produce generally accurate outputs, but not as accurate as the other three models. For illustration purposes, the outputs of these two models are compared to their counterparts in the test data in Figs. 6.3 and 6.4 in both time and frequency domains. The results indicate that the peak frequencies are captured by the model, including the first tower FA bending mode at ≈ 0.88 Hz. As to model formulations, the DTM forms are slightly more accurate than the SM form for Ω, ω, and P but are similar for the other cases. The overall accuracy of SM and DTM is significantly different only in representing P on training data, according to a Wilcoxon rank sum test ($p = .041$).

The performance of global models is summarized in Table 6.3, where their accuracy is compared with other models commonly reported in the literature. Specifically, the accuracy of SM and DTM forms is compared to that of models in the form of multiple regression and 20^{th} order ARX models. The accuracy of the DTM-LA form is compared to that of a NARX-NN. The results in Table 6.3 indicate excellent accuracy of the models of Ω, ω, and P obtained by ELGP, and the lower accuracy of the models of M_{FA} and M_{SS}, as was also observed with the local models in Table 6.2. As with the local models, a comparison of the outputs of these models to data in both time and frequency domains, shown in Figs. 6.5 and 6.6, indicates accurate representation of low-order dynamics and the peak frequencies. The results in Table 6.2 also indicate the better accuracy of the SM and DTM models generated by ELGP than that of the linear regression or the ARX model forms. The accuracy difference is particularly pronounced in the M_{FA} and M_{SS} models as represented by the 45-50% higher VAF values and the models of Ω and ω as characterized by the higher VAF values of approximately 10-28%. As to the performance of the DTM-LA form, which is compared to that of a NARX-NN, the tower moment predictions, M_{FA} and M_{SS}, are significantly better than those of SM and DTM by both methods. Although ELGP and NARX-NN show

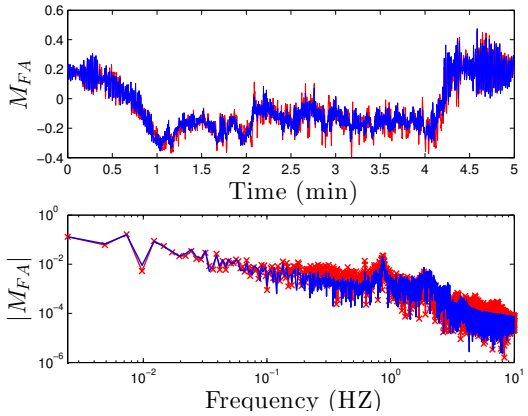

Figure 6.3. SM M_{FA} model (blue) with training and validation data (red) at $\bar{V} =$ 16.0 m/s.

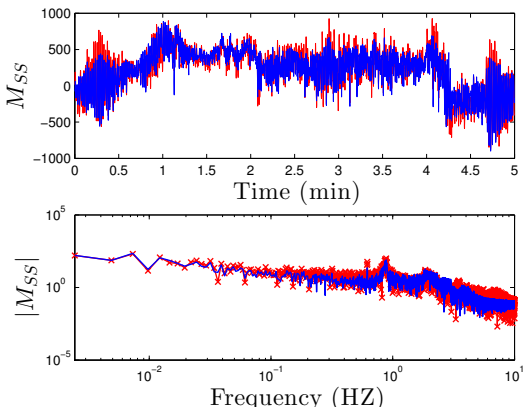

Figure 6.4. SM M_{SS} model (blue) with training and validation data (red) at $\bar{V} =$ 16.0 m/s

nearly identical prediction capability in all cases, they differ in transparency (intelligibility), as discussed next.

6.8.1 Model Interpretation

ELGP maintains an archive of solutions that are nondominated with respect to the objectives of fitness and complexity during identification for the purpose of providing less complex and possibly more general alternatives to the best training solutions. Previous research has suggested that models with physical insight normally reside along the edges of the Pareto set, where a small increase in complexity could result in large improvement in estimation accuracy [177]. We observe this phenomenon in our results, as shown in several of the archives in Figs. 6.7 - 6.10, with "n" being a placeholder for all constants.

6.8.1.1 Local Models

We find that the low complexity models in the archives often identify basic relations in the closed-loop system. For example, consider the local models created by ELGP as illustrated in Figs. 6.7– 6.9, which show the models on the Pareto front of the archives. The first figure, Fig. 6.7, corresponds to the models of ω in SM form in below-rated operating conditions where the torque control strategy is $T_G = k\omega^2$. Low-complexity solutions include $\omega = \frac{n}{T_G}$,

Table 6.2. [Performance of local models] Performance of local models generated by ELGP using SM and DTM model formulations. Results are categorized by the mean wind speed ($\bar{V}$) of the corresponding 5 minute data set.

	Training/Validation VAF (%)				
$\bar{V}$ (m/s)	Ω	ω	M_{FA}	M_{SS}	P
			SM		
4.5 m/s	99.3 / 99.2	99.4 / 99.4	97.5 / 97.3	95.1 / 94.5	99.8 / 99.7
7.1 m/s	97.0 / 96.9	98.0 / 97.9	80.4 / 79.1	86.6 / 85.4	99.9 / 99.9
7.9 m/s	99.9 / 99.9	99.8 / 99.8	80.2 / 79.1	85.8 / 85.6	100.0 / 100.0
8.9 m/s	99.9 / 99.9	99.8 / 99.8	90.5 / 90.3	89.9 / 89.3	100.0 / 100.0
9.5 m/s	99.6 / 99.6	99.3 / 99.3	83.8 / 83.7	81.9 / 81.1	99.8 / 99.8
10.6 m/s	99.8 / 99.9	99.7 / 99.7	91.3 / 90.9	87.8 / 87.7	99.8 / 99.8
10.9 m/s	99.7 / 99.7	99.7 / 99.6	92.0 / 92.0	82.7 / 82.7	99.8 / 99.8
12.3 m/s	98.6 / 98.5	98.1 / 98.0	79.0 / 78.9	73.2 / 71.8	99.4 / 99.4
14.2 m/s	99.9 / 99.8	99.8 / 99.8	95.9 / 95.9	73.2 / 73.4	99.8 / 99.8
16.0 m/s	99.1 / 99.1	99.0 / 99.1	94.2 / 94.2	85.0 / 83.5	99.9 / 99.9
16.3 m/s	97.8 / 97.8	97.1 / 97.0	71.8 / 71.0	82.7 / 82.6	99.6 / 99.5
17.0 m/s	99.6 / 99.6	99.5 / 99.5	83.3 / 83.0	83.8 / 83.6	99.9 / 99.9
18.0 m/s	18.2 / 17.0	21.8 / 22.3	52.7 / 51.0	70.2 / 70.8	72.3 / 72.4
18.9 m/s	99.9 / 99.9	99.9 / 99.9	92.3 / 92.7	96.6 / 97.0	100.0 / 100.0
			DTM		
4.5 m/s	99.4 / 99.4	99.4 / 99.3	97.6 / 97.4	95.3 / 95.3	99.8 / 99.8
7.1 m/s	99.2 / 99.2	99.4 / 99.3	80.8 / 80.9	93.2 / 92.1	99.9 / 99.9
7.9 m/s	100.0 / 100.0	100.0 / 100.0	88.2 / 89.1	92.3 / 92.0	100.0 / 100.0
8.9 m/s	100.0 / 100.0	100.0 / 100.0	97.1 / 96.9	95.9 / 96.0	100.0 / 100.0
9.5 m/s	99.8 / 99.8	99.7 / 99.7	91.3 / 90.2	85.6 / 85.1	99.9 / 99.9
10.6 m/s	99.9 / 99.8	99.8 / 99.7	92.3 / 92.7	90.4 / 89.5	100.0 / 100.0
10.9 m/s	99.8 / 99.8	99.6 / 99.6	91.7 / 90.8	87.0 / 86.8	100.0 / 100.0
12.3 m/s	98.8 / 98.8	98.3 / 98.3	80.8 / 79.7	81.4 / 81.0	99.8 / 99.8
14.2 m/s	99.8 / 99.8	99.9 / 99.9	95.9 / 96.0	76.7 / 76.5	99.9 / 99.9
16.0 m/s	99.1 / 99.0	99.1 / 99.0	94.2 / 93.8	86.5 / 86.6	100.0 / 100.0
16.3 m/s	97.9 / 98.1	97.1 / 97.6	73.4 / 74.3	86.1 / 83.6	99.9 / 99.9
17.0 m/s	99.5 / 99.5	99.5 / 99.5	85.3 / 85.8	86.2 / 86.5	100.0 / 99.8
18.0 m/s	90.0 / 63.8	81.5 / 56.1	61.1 / 60.1	78.5 / 77.4	76.4 / 78.1
18.9 m/s	100.0 / 99.9	99.9 / 99.9	91.9 / 91.1	97.3 / 96.8	100.0 / 100.0

Table 6.3. Comparison of global models generated by ELGP (SM, DTM, DTM-LA), two linear system identification methods (multiple regression, ARX), and a neural network (NARX-NN). The one-step prediction models (DTM-LA and NARX-NN) are grouped on the right. The best method for each case is in bold.

	Training/Validation VAF (%)					
	SM	DTM	Multiple Regression	20^{th}-order ARX	DTM-LA	NARX-NN
Ω	98.4 / 96.9	**98.7 / 98.7**	91.9 / 91.9	71.0 / 71.0	**100.0 / 99.9**	99.9 / 99.8
ω	97.8 / 98.4	**98.6 / 98.6**	92.0 / 91.9	69.0 / 69.0	**100.0 / 99.9**	99.9 / 99.8
M_{FA}	**76.0 / 76.1**	74.2 / 74.4	31.5 / 32.2	25.6 / 25.6	**98.7 / 94.9**	98.6 / **94.9**
M_{SS}	69.5 / 69.6	**72.7 / 72.2**	19.6 / 20.4	0.0 / 0.0	**97.6** / 89.9	97.3 / **90.6**
P	**99.9 / 99.9**	**99.9 / 99.9**	99.7 / 99.7	99.6 / 99.6	- / -	- / -

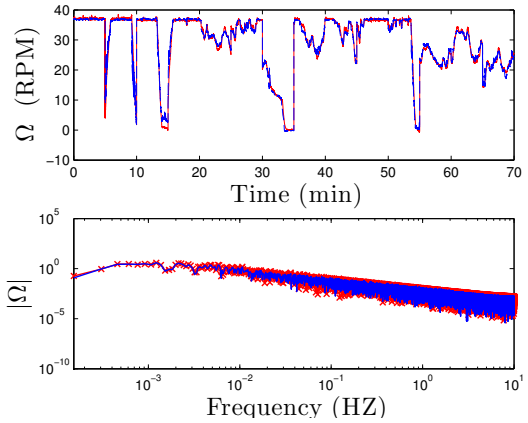

Figure 6.5. Comparison of the global Ω SM model (blue) and combined training and validation data (red).

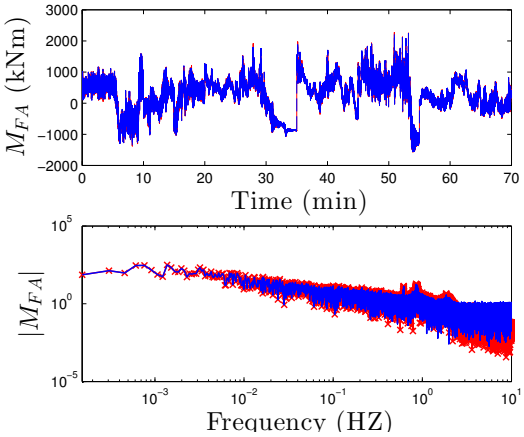

Figure 6.6. Comparison of the global DTM-LA M_{FA} model (blue) and combined training and validation data (red).

which is the analytical solution to the power law $P = \omega T_G$ when P is constant, and $\omega = (n - T_G)T_G = nT_G - T_G^2$, which in the absence of square root or exponent operators, bears some resemblance to the Taylor series approximation for square root: as $\sqrt{1+x} \approx 1 + \frac{1}{2}x - \frac{1}{8}x^2$, characterizing the relationship $\omega = n\sqrt{T_G}$. The basic closed-loop relationship between ω and β is also identified in DTM form in above-rated operation, as shown in Figure 6.8; in this case, a low-complexity model on the edge of the Pareto set is $\omega_k = n(\omega_{k-1} + (\beta - \beta_{k-1}))$, which describes the proportional control effort of β with respect to ω and generalizes better than more complex models. The archives of local power models also contain process physics, as shown in Figure 6.9. The relation $P = \omega T_G$ occupies the elbow of the curve, with slight variations of this increasing model accuracy at the cost of complexity. An interesting solution scales the power law by a nonlinear function of the wind disturbance: $P = \omega T_G / \sin(e^{V/n})$.

We are able to draw two main insights from the local model archives. First, the feature selection property of GP results in local models that describe the local closed-loop system without inactive control input variables. This is evident in the comparison of the below-rated models of ω (Figure 6.7) that depend mostly on T_G and the above-rated models (Figure 6.8) that depend mostly on β. Second, the appearance of the control strategies in the low-

complexity models of the archive indicates that the closed-loop dynamics exhibited by the wind turbine are in some cases heavily defined by the controller behavior.

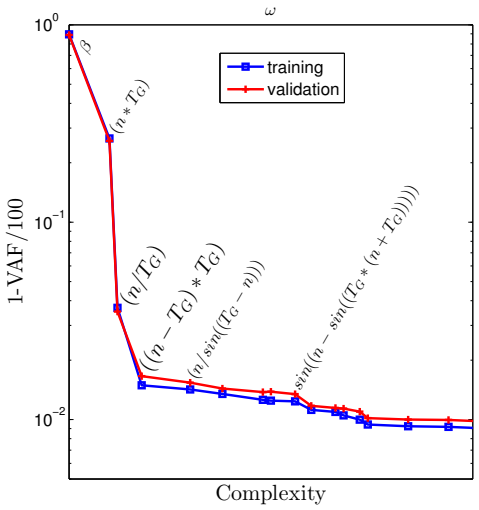

Figure 6.7. Pareto archive of SM models of ω at $\bar{V} = 9.5$ m/s.

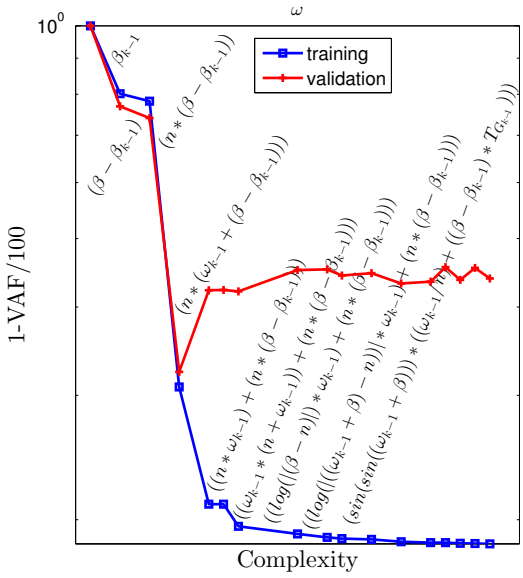

Figure 6.8. Pareto archive of DTM models of ω at $\bar{V} = 18.0$ m/s.

6.8.1.2 Global Models

Performing the model search globally has the advantage of testing candidate models across control regimes that vary in their relation to the turbine behavior; this may help distinguish the behavior of the wind turbine from the changing relations governing the inputs. Indeed, we find that the global DTM and DTM-LA models are generally more dependent on several inputs, including their previous outputs and V, which indicates that the behavior of the plant is more uniquely identified. For example, Figure 6.10 shows the Pareto archive of Ω from global identification. The final model shown is a nonlinear function of V_{k-1}, β_{k-1}, $T_{G_{k-1}}$, and Ω_{k-1}. It contains the same term $\sin(e^{V/n})$ found in the local power models, suggesting that this may be a concise way for models to represent the nonlinear response of outputs to V.

The global identification of P converges within eight generations on the power law, $P = \omega T_G$, which remains the most accurate global model found of any complexity. Interestingly,

the success of the linear predictions of P in Table 6.3 can be understood using the archive of global P solutions that contains only two nondominated solutions: $P = nT_G$ and $P = nT_G\omega_n$. Hence, the ELGP archive contains a linear model of power based on T_G that has approximately the same VAF value as the linear models (99.2%). Note that the ELGP models are more parsimonious than those generated by the linear methods due to feature selection.

The results in Table 6.3 show that the ELGP method is able to produce one-step look-ahead models that are as accurate as those produced by a black-box neural network model. Compared to the NARX-NN models that are undecipherable, those generated by ELGP are quite succinct:

$$\widetilde{\Omega}_k = \widetilde{\Omega}_{k-1} - \sin(\frac{n_1}{t}) \sin\left(\frac{n_2 \widetilde{V}_{k-1} \widetilde{T}_{G_{k-1}}}{\widetilde{\Omega}_{k-1}(\beta_{k-1} + n_3)}\right) \tag{6.9}$$

$$\widetilde{M_{FA}}_k = \widetilde{M_{FA}}_{k-1} + n_1 \sin(\widetilde{T_G})(\tilde{\ddot{x}}_{FA} - \tilde{\ddot{x}}_{FA_{k-1}})/\widetilde{V} \tag{6.10}$$

$$\widetilde{M_{SS}}_k = \widetilde{M_{SS}}_{k-1} + n_1 \sin(n_2\psi)(\tilde{\ddot{x}}_{FA} - \tilde{\ddot{x}}_{FA_{k-1}}) \tag{6.11}$$

Note that $(\tilde{\cdot})$ denotes the scaled variables. In each case the model consists of the summation of the previous output and a compact nonlinear function. For the tower moment cases, the nonlinear component contains the change in fore-aft acceleration of the tower top ($\tilde{\ddot{x}}_{FA} - \tilde{\ddot{x}}_{FA_{k-1}} = \Delta\tilde{\ddot{x}}_{FA}$), which is an intuitive result. Interestingly, the model for M_{SS} decomposes $\Delta\tilde{\ddot{x}}_{FA}$ into its side-side component using the yaw angle, i.e., $\sin(n_3\psi)$, rather than using the measurement of $\ddot{x}_{SS}$. This may indicate that $\ddot{x}_{FA}$ is more reliably measured than $\ddot{x}_{SS}$.

6.9 Discussion

The local and global models obtained by ELGP using EMO demonstrate the potential for the succinct resultant models to enhance the understanding of the characteristics observed from a process. We have shown that the models are transparent enough to link process estimation to understandable model components (Eqns. (6.9–6.11)) and accurate enough to

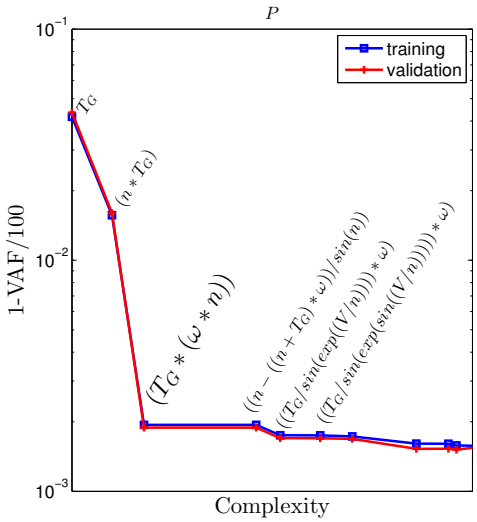

Figure 6.9. SM Pareto archive of P at $\bar{V} = 7.1$ m/s.

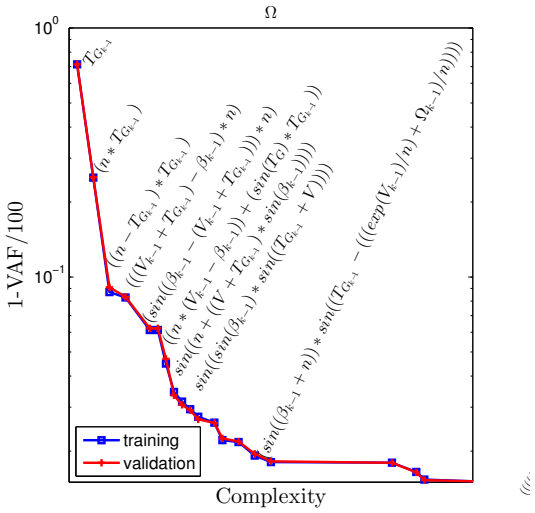

Figure 6.10. DTM Pareto archive of global models of Ω.

shed light on the characteristics of the closed-loop behavior of the system. Furthermore, by studying the archive we are able to see how specific increases in complexity affect the model's estimation capacity. In general, the models do not suffer from overfitting (with some exceptions, e.g., Figure 6.8 and the DTM-LA tower moment models). This may be due to the way in which the validation set is chosen or be a property of the model forms that are discovered.

The produced models hold a utility beyond the drawing of insight into the process that is modeled. There is clearly a need for high-fidelity local models of wind turbines for control design [9]. The models can be used directly in a model predictive control system, or linear models of the wind turbine at different wind speeds can be derived if the control design process requires them. These models could be generated from the global nonlinear models by several approaches. One approach is to linearize the model forms analytically; another is to build auto-regressive models from the output of the nonlinear model in quasi-linear operating regimes. Deriving the linear models directly from the predicted output of the model may provide more flexibility in choosing the structure of the linear model. Even the static models may be utilized as a target for the transformation to a set of local linear dynamic models.

A direct comparison to previous CART3 identification results is difficult given the difference in operating conditions and assumptions, although the results here compare well to those previously published [207] that use perturbance injection. It is likely that the results could be further improved if persistent excitation signals were used or the model formulations were higher than first order. However, we have shown that accurate models can be obtained even with the limited experimental setup we employ. We have chosen to use these restrictions to make the results relevant to data collected from wind turbines during regular operation, which is far more common, and requires less expertise to obtain. The limited assumptions regarding the aerodynamic properties of the turbine broadens the applicability of the work to poorly characterized turbines as well. In terms of computation time, this approach is more efficient than standard GP but incurs a higher cost than linear methods. Typical runs over the global data for DTM model training require several hours to converge. Fitness estimation methods [175] can assist in scaling as the dimensionality of the data set increases.

6.10 Conclusion

In this work we use a novel symbolic regression system in an evolutionary multiobjective optimization framework to identify compact models of a wind turbine from operating data with minor assumptions. The models are not only accurate, but succinct and intuitive, and have been shown to embody process knowledge in several instances. This method of system identification may be a promising middle ground between conducting computationally expensive physics simulations and using black-box models since the models evaluate quickly while still capturing the intelligible system behavior. In the future we plan to fully characterize how the sets of assumptions about the aerodynamic properties of the turbine and the operating conditions change the fidelity of the identified models.

As wind turbines continue to grow in size and flexibility and begin to move offshore onto floating platforms, we expect data-based modeling of wind turbine behavior to become an even more integral part of the design and research processes. The use of intelligible modeling methods can help catalyze expert knowledge of turbine behavior in response to combined

wind and wave loading. Even small gains in power capture of utility-scale wind farms can result in large financial gains, and therefore it is crucial to improve the power capture as well as the lifetime and maintenance of these machines. For this reason, we expect that data-based approaches such as this one are key to continuing the success of wind energy technology worldwide.

6.11 Acknowledgments

I would like to thank Dr. van der Veen for sharing his insights into identification of wind systems. This work was also supported by the U.S. Department of Energy under Contract No. DE-AC36-08GO28308 with the National Renewable Energy Laboratory. Funding for the work was provided by the DOE Office of Energy Efficiency and Renewable Energy, Wind and Water Power Technologies Office.

CHAPTER 7

AGENT-BASED DYNAMIC MODELING OF BALD EAGLES NEAR WIND FARMS

7.1 Summary

This chapter considers the identification of dynamic behavior models of bald eagles that combine multiclass classification and regression. Several machine learning approaches are considered in addition to M4GP and ELGP, as defined in Chs. 3 and 4, respectively. The data set consists of time series measurements of bald eagle locations in Maine, in addition to features based on proximity to geographic features that may affect their observed decision making. We find accurate models to predict bald eagle behavior based on previous time-steps using M4GP as well as decision tree classification. The task of accurately predicting flight length and direction remains unattainable with respect to the current methodology.

7.2 Introduction

Bald eagles (*haliaeetus leucocephalus*) are the national bird of the United States, and are considered a culturally and nationally important species. Maine is home to a large bald eagle population (see Figure 7.1), and is also a critical renewable energy state in the Northeast due to its strong wind resource. Because bald eagle behavior is not well understood, regulation governing the sitings of new wind farms fails to address the impact on this species in an informed way. To improve this understanding, researchers in the Ecological Conservation department at UMass are collecting data from bald eagles in Maine by instrumenting them with GPS sensors that provide location information every 15 minutes in real time. The goal of project is to model the behavior of these birds, especially in response to man-made and natural geographic features, so that regulatory decisions can be better informed. These models must be interpretable so as to inform researchers and policy makers in the field.

To simulate the temporal (i.e. causal) behavior of bald eagles, dynamic models are required. Dynamic models make predictions sequentially based on information from previous states; this previous information may come from previous model predictions in fully virtual environments, or may come from previous measured states, as I assume in this paper. I use dynamic modeling formulations of several machine learning (ML) methods in this work, including Naïve Bayes, LASSO, decision trees (DT) and genetic programming (GP), detailed in §7.4. Models are created both to classify bald eagle behavior and predict their movement, as shown in §7.6 and discussed in §7.7. The results suggest DT- and GP- based models are able to learn accurate behavior classifiers that are human-readable. For the task of predicting bird movement, however, the methods tested fail to produce dynamic models that correlate well with test data. Among the dynamic modeling methods, LASSO and the GP methods produce similar results, whereas the DT method makes uncorrelated predictions on the test set.

7.3 Related Work

Dynamic models are actively studied for system identification in engineering applications [121, 12] and are the basis of most model-predictive control systems [4]. Auto-regressive forms of some ML algorithms have been proposed [12], including dynamic bayesian networks [142] and autoregressive decision trees [135] which are used here. Murphy [144] showed that dynamic bayesian networks generalize hidden Markov models and Kalman filter models by relaxing some assumptions of those methods. It should be noted that the dynamic formulation of Lasso matches the auto-regressive exogenous input (ARX) model form [121] except for the $\ell 1$ regularization penalty.

The application of these modeling techniques to bald eagle behavior is novel, although agent-based models of birds have been proposed. For example Abden [1] modelled the movement of Afrotropical forest birds using a stochastic movement simulator customized to match field observations. Machine learning, specifically DTs, have been used in other areas of avian ecology: for example Massey [131] used DTs to classify the nesting habitat of Norther Harriers.

7.4 Data Set

The current data set consists of 37340 samples (N_1) collected from 17 birds, with 25 features measured at each time step, including latitude, longitude, and altitude, distance to various landscape features such as water, and distance to nest and other eagles. For the purposes of modeling, the feature vector $\mathbf{x} \in \mathbb{R}^D$ is reduced to the following 10 relevant attributes: step length; turn angle; direction to nest; distance to nearest eagle; home distance; distance to edge of territory; distance to nearest body of water; direction to nearest body of water; altitude; and land cover classification. In addition, experts in the field have labeled the behavior of these birds into 4 categories: *perch* (the bird sitting in a tree), *flight* (typical for foraging), *nest* (the bird sitting in its home tree), and *cruise* (a territorial flying behavior at high altitude). The goal of the project is to build a model that predicts not only which behavior the eagle will exhibit, but, in the case of mobile behaviors (*flight* and *cruise*), where

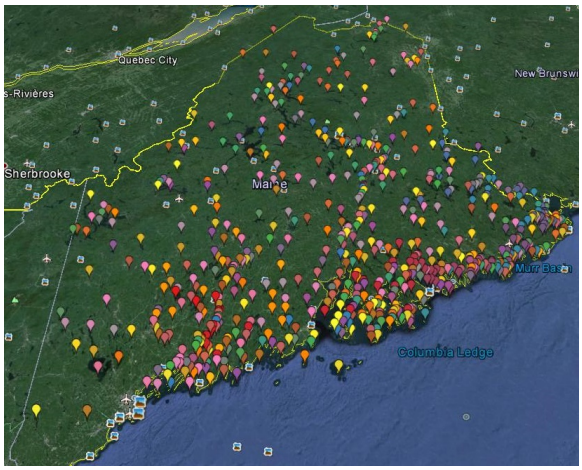

Figure 7.1. Bald eagle nest locations recorded in Maine.

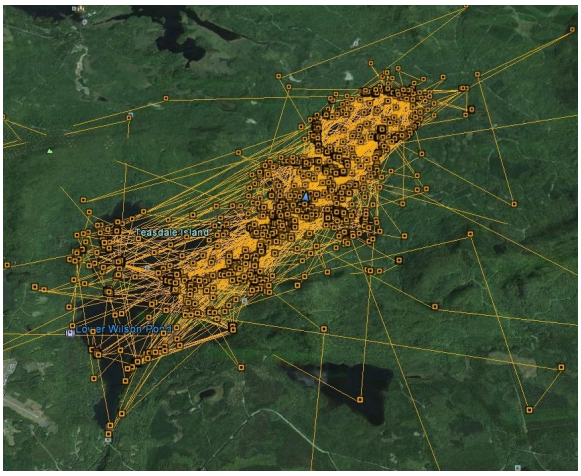

Figure 7.2. GPS locations of a single bald eagle in the data set.

the bird will fly in terms of flight distance (step length) and direction (turn angle). 2420 samples (N_2) include mobile behaviors. A key assumption of this system is that the eagle makes decisions based on k previous states (i.e. memory).

7.5 Proposed Solution

The problem can be formulated as a two-stage modeling challenge: the first stage is to appropriately predict the eagle behaviors $y^c \in \mathcal{C} = \{perch,\ flight,\ nest,\ cruise\}$ via a kth-order dynamic function

$$f\left(\mathbf{x}_{i-1}, \ldots, \mathbf{x}_{i-k}, y^c_{i-1}, \ldots, y^c_{i-k}\right) : \mathbb{R}^{D_1} \rightarrow \mathcal{C}$$

trained on the labeled set $\mathcal{D}^c = \{(\mathbf{x}_i, y^c_i)\}_{i=1}^N$, where y^c_{i-1} is the class of the previous timestep. The second stage is to find two real-valued, kth-order dynamic functions representing step length, y^s, and turn angle, y^a, of the form

$$f\left(\mathbf{x}_{j-1}, \ldots, \mathbf{x}_{j-k}, y^s_{j-1}, \ldots, y^s_{j-k}, y^a_{j-1}, \ldots, y^a_{j-k}\right) : \mathbb{R}^{D_2} \rightarrow \mathbb{R}$$

that predict the planar movements of the eagles when flying or cruising, i.e. from the training sets $\mathcal{D}^s = \{(\mathbf{x}_j, y_j^s)|y_j^c = cruise \vee flight\}_{j=1}^{N_2}$ and $\mathcal{D}^a = \{(\mathbf{x}_j, y_j^a)|y_j^c = cruise \vee flight\}_{j=1}^{N_2}$.

In these experiments, first-order dynamic systems are considered, i.e. $k = 1$. For simplicity, I denote the generic feature vector constructed from a single previous timestep's measurements and output as $\tilde{\mathbf{x}}$, such that $\tilde{\mathbf{x}}_i = [\mathbf{x}_{i-1}\ \mathbf{y}_{i-1}]$. In the case of classification, $\tilde{\mathbf{x}}$ has D_1 dimensions; for regression of step length and turn angle, $\tilde{\mathbf{x}}$ has D_2 dimensions.

In order to normalize for the largely different scaling of the measurement data, the raw data was pre-processed using standard scaling techniques: real-valued features were normalized to 0 mean, unit variance, and categorical data (land cover) and class labels were normalized to integers.

7.5.1 Analysis of feature space

As a first step in exploratory analysis, the natural structure of the feature space is studied using k-means clustering. The number of clusters and their locations are compared to the class labels in order to determine 1) whether the class labels assigned by the researchers have well-formed boundaries in the feature space and 2) whether the number of behavior classifications match the natural clusters. k-means is a coordinate descent method that relies on updating the cluster assignments of the data cases and centroids of the clusters in two distinct steps. To begin, k centroids μ are initialized, and then the following two steps are repeated:

1. update z_i, the cluster assignment of sample $\tilde{\mathbf{x}}_i$, as $z_i = \text{argmin}_k\, ||\mu_k - \tilde{\mathbf{x}}_i||_2^2$
2. update the cluster centroids based on the new assignments as $\mu_k = \frac{\sum_{i=1}^{N} [z_i=k]\tilde{\mathbf{x}}_i}{\sum_{i=1}^{N} [z_i=k]}$

Steps 1. and 2. are repeated until convergence. I selected k-means for its compact representation of clusters and its low computational complexity ($O(TNDK)$ for T iterations) compared to other methods. It has the guarantee of non-increasing iterative updates to its objective function, defined as

$$\sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{\mathbf{x} \in C_k} ||\mathbf{x} - \mu_k||_2^2 \tag{7.1}$$

where C_k defines a cluster k. I use k-means++ initialization, which initializes centroids to the data points furthest apart. To find the natural number of clusters in the data, k is varied from 2 to 9 and the average silhouette score of the resulting clusters is calculated, such that the optimal number of clusters can be chosen via the elbow method.

7.5.2 Classification of bald eagle behavior

7.5.2.0.1 Decision Tree Classification DT classification makes predictions using a conjunction of rules into a binary tree structure. Beginning with all the training dat at the root, internal nodes of the tree split data according to greedily computed threshold values on single features, and a given leaf predicts the dominant class of the subset of $\mathcal{D}$ that reaches that leaf when routed through the tree. DTs are chosen for their legibility, as well as their ability to make non-linear predictions and also balance bias and variance by tuning the maximum tree depth. DTs also implicitly conduct feature selection by choosing features for splits based on subsets of $\mathcal{D}$. However, as a weakness, it only makes axis-aligned distinctions in the data, which can be slow for non-axis aligned case splits. The hyperparameters of max depth ([2:5]), split type ([best, random]) and split criteria ([gini,entropy]) are tested.

7.5.2.0.2 Naïve Bayes NB is an approximation of the Bayes optimal classifier that assumes all data dimensions are independent and follow a chosen distribution, in this case Gaussian. The first order dynamic NB classifier assigns labels according to the maximum probability of a class label given the previous state, $P(Y = c|X = x)$, according to (a non-normalized) Bayes rule, as

$$f_{NB}(\tilde{\mathbf{x}}) = \underset{c \in \mathcal{C}}{\operatorname{argmax}}\ (y_i = c|\tilde{\mathbf{x}}) = \underset{c \in \mathcal{C}}{\operatorname{argmax}}\ \pi_c \phi_c(\tilde{\mathbf{x}}_d)$$

where $\pi_c = P(y_i = c)$ and ϕ_c is the probability of observing $\mathbf{x}_{i-1}$ and y_{i-1} succeeded by behavior c. By assuming independence of features, ϕ_c can be modelled as the product of independent distributions ϕ_{cd}, as $\phi_c(\tilde{\mathbf{x}}) = \prod_{d=1}^{D_1} \phi_{cd}(\tilde{\mathbf{x}}_d)$. NB relies on the user to specify the distribution to use for ϕ_{cd}, which is the Gaussian distribution here. The learned parameters are the mean and variance of each attribute.

7.5.2.0.3 Multidimensional Genetic Programming

Multidimensional genetic programming [188, 140] searches for feature transformations and succinct model forms using a variant of genetic programming (GP) [93]. The goal is to to find a set of transformations $\Phi(\tilde{\mathbf{x}}) : \mathbb{R}^D \rightarrow \mathbb{R}^p$ that project $\tilde{\mathbf{x}}$ into a p-dimensional space in which the samples are more easily classified according to their distribution distances. The GP system optimizes the following problem:

$$\Phi^*(\tilde{\mathbf{x}}) = \arg\max_{\Phi \in \mathbb{S}} F_1\left(\Phi, \mathcal{D}\right) \;;\; F_1 = \sum_{\ell}^{|\mathcal{C}|} 2\frac{\text{precision}_\ell \cdot \text{recall}_\ell}{\text{precision}_\ell + \text{recall}_\ell} \tag{7.2}$$

where $\mathbb{S}$ is the space of possible transformations, Φ. The F_1 measure (used here as the GP fitness function) defines classifier accuracy in terms of precision precision ($\frac{TP}{TP+FP}$) and recall ($\frac{TP}{TP+FN}$), where TP are true positive classifications, FP are false positives, and FN are false negatives. Φ is a set of syntax trees consiting of mathematical building blocks and features, optimized via genetic programming operations (selection and variation). To classify a sample, $\mathcal{D}$ is partitioned according to its class labels into matrices $\{\tilde{\mathbf{X}}_1 \; \ldots \; , \tilde{\mathbf{X}}_{|\mathcal{C}|}\}$ and the minimum distance of $\Phi(\tilde{\mathbf{x}})$ to each transformed subset determines the class assignment, i.e.

$$\hat{y}(\Phi(\tilde{\mathbf{x}})) = c_j, \;\; \text{if} \;\; j = \arg\min_{\ell=1}^{|\mathcal{C}|} d\left(\Phi(\tilde{\mathbf{x}}), \Phi(\tilde{\mathbf{X}}_\ell)\right) \tag{7.3}$$

The Mahalanobis distance is used for d. The implementation considered for this project[1] incorporates advanced parent selection techniques and is known as M4GP [108]. The main hyperparameters for M4GP are the population size ([100, 500]) and the maximum size of the programs representing Φ, in terms of numbers of nodes ([20, 50]).

7.5.3 Prediction of bald eagle movement

7.5.3.0.4 Decision Tree Regression

DT regression is very similar to DT for classification (§7.5.2.0.1), except that the leaves predict the mean of y among the subset of cases routed to that leaf. Node thresholds are chosen by greedily minimizing the variance of the

[1] code available for M4GP and ELGP are available from http://github.com/lacava/ellen

outputs of the subsets created by splitting $\mathcal{D}$ at the threshold value. The hyperparameters of max depth ([2:11]) and split type ([best, random]) are tested.

7.5.3.0.5 Lasso Lasso is a linear regression method that adds an ℓ_1 regularization penalty to the loss function, as:

$$\mathbf{w}_* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \alpha ||\mathbf{w}||_1$$

Here, $\hat{y} = \mathbf{w}^T \tilde{\mathbf{x}}$ is the prediction of $f(\tilde{\mathbf{x}})$ on $\mathcal{D}$. The strength of Lasso is that the ℓ_1 regularization implicitly conducts feature selection by pressuring coefficients to be zero. It is also fast: the hyperparameter of Lasso is α, but actually the full regularization path (i.e. the value of α that produces the best result) can be computed efficiently using least-angle regression (LARS), in which features are incrementally added to the model based on their angle with respect to the current residual. I used `LassoLarsCV` to automatically tune α.

7.5.3.0.6 ELGP Epigenetic linear genetic programming is a method proposed for system identification [107, 105] that optimizes a population of programs that represent models of the system using fitness-proportionate selection and bio-inspired search operators. Unlike traditional GP approaches, ELGP uses a unique stack-based representation with epigenetic markers (on/off conditions) on each node in the programs that are optimized each generation via stochastic hill climbing. Like M4GP (§7.5.2.0.3), the main hyperparameters are the population size ([100, 500, 100]) and the maximum program size ([10, 20]).

7.6 Experiments and Results

5 clusters were found to maximize the average silhouette score using k-means, compared to 4 classes assigned to behavior by the researchers. I find that one of the k-means clusters does have very few data samples (not shown here), suggesting that 4 clusters cover most of the data cases. The 5 cluster centers are plotted along the first two principal component axes (the directions of maximum variance in the data set) in Figure 7.3. The data is colored according to its class label. Although the k-means cluster centers are well separated along the axes of highest variance, and 4 cluster centers appear to cover most of the data, it is clear

that the principal variance directions in the data do not correspond to the class labels of the data, which are heavily mixed along the axes.

The data was divided 50/50 into training and test sets. The hyperparameters of the models from the scikitlearn implementations were optimized using 5-fold cross validation over the sets of hyperparameters defined in §7.5. The GP methods (M4GP and ELGP) were optimized using 5-sample random resampling and test. The best hyperparameters for each method were then used to train the final models on the entire training set. These models then were used to predict the labels and outputs for the test set. Classification accuracy is compared in terms of the F_1 measure (Eq. (7.2)); regression accuracy is compared in terms of the coefficient of determination, R^2.

Figures 7.5 - 7.7 show the accuracy of the studied methods. Figure 7.5 shows that DT ($F_1 = 0.882$) and M4GP ($F_1 = 0.873$) generate similarly accurate predictions of bald eagle behavior, and NB produces a less accurate model ($F_1 = 0.6988$). The cross-validated DT model has a depth of 3, rendering a legible model shown in Figure 7.4. In comparison, M4GP produces a model with 6 feature transformations of less than 3 nodes using 3 features. Figures 7.6 and 7.7 show the relatively low accuracy of the regression models for predicting bald eagle movement ($R^2 < 0.2$, 0.002). The DT methods do not generalize for either case; Lasso and ELGP produce similar models for step length, and ELGP produces a slightly more accurate model for turn angle.

7.7 Discussion and Conclusions

The classification results indicate that DT and M4GP can produce accurate dynamic classifiers of bald eagle behavior data. The DT model in Figure 7.4 is being shared with Environmental Conservation researchers to assist in understanding eagle behavior; it shows in particular a change in behavior as the eagles move far from their nesting location, and no sensitivity to geographic features. The dynamic models of bald eagle movement do not have significant predictive power. Further work should address whether higher-order models can

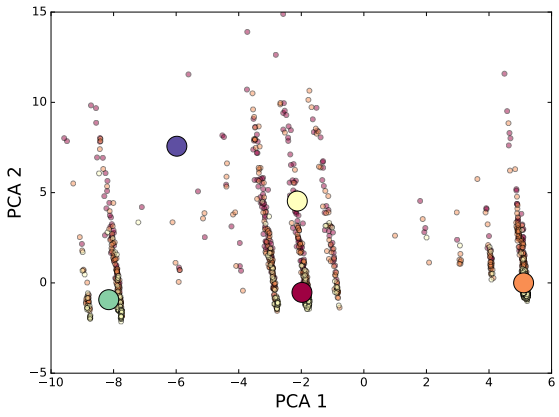

Figure 7.3. k-means cluster centers (circles) and training data (dots) along the first two principal component axes. The training data is colored according to its true class label.

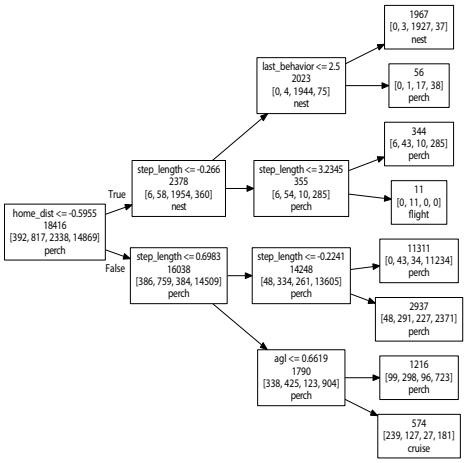

Figure 7.4. Cross-validated decision tree classifier for bald eagle behavior.

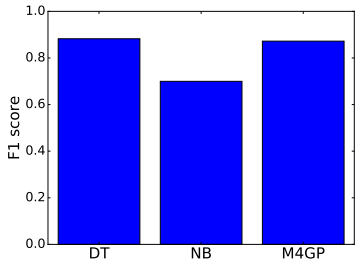

Figure 7.5. F_1 scores for behavior on the test set for each method.

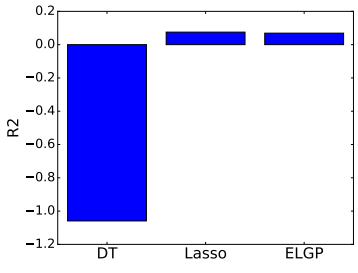

Figure 7.6. R^2 scores for step length on the test set for each method.

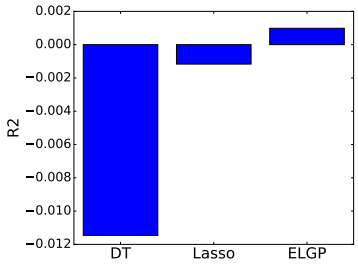

Figure 7.7. R^2 scores for turn angle on the test set for each method.

improve the accuracy, or whether the measurement time step (15 minutes) is small enough to capture causality.

CHAPTER 8

TOWARDS GLOBAL MODELING OF VORTEX-INDUCED VIBRATIONS ON CYLINDRICAL STRUCTURES

8.1 Summary

In this chapter, an approach to consolidating local models generated by ELGP is proposed. As a case study, I consider the identification of vortex-induced vibration (VIV) on a cylindrical structure, subject to varying flow velocities. This process exhibits qualitatively different behavior for different flow velocities due to the frequency-dependent interactions between the fluid and the cylinder. To produce generalized models using ELGP, local models are used to seed the initial population of a globally-trained model. In addition, I implement a novel parent selection scheme for regression, known as ϵ-lexicase selection, that differs from traditional GP search divers in that it propagates models that are accurate on unique combinations of training cases. Simulations of the resultant model are compared to measurement data, showing accurate reproduction of the phase changes between cylinder displacement and the fluid force as flow velocity changes. We discuss modeling challenges with respect to this system and provide a road map of future research.

8.2 Introduction

Generalization approaches are needed to adequately model systems for which local behaviors are dominated by differing aspects of the process, leading to poor generalization of locally trained models. One way to improve the generalization performance of GP on difficult problems is to seed the initial population with solutions from simpler problems [112, 179]. Still, when performing global identification on the cascaded training sets, there is a high likelihood that local solutions that perform well on specific regimes of operation will be lost, due to inaccuracies in other conditions. This loss is due to the way in which GP reduces a program's performance into a single value that is used to select parents for the next generation. Typically the fitness F of an individual is quantified as its aggregate performance over the training set $\mathcal{T} = \{(y_t, \mathbf{x}_t)\}_{t=1}^{N}$, using e.g. the mean absolute error (MAE), which is quantified for individual program $i \in P$ as:

$$F(i, \mathcal{T}) = \frac{1}{N} \sum_{t \in \mathcal{T}} |y_t - \hat{y}_t(i, \mathbf{x}_t)| \tag{8.1}$$

where $\mathbf{x} \in \mathbb{R}^D$ represents the variables or features, the target output is y and $\hat{y}(i, \mathbf{x})$ is the program's output. As a result of the aggregation of the absolute error vector $e(i) = |y - \hat{y}(i, \mathbf{x})|$ in Eq. (8.1), the relationship of $\hat{y}$ to y is represented crudely when choosing models to propagate. As others have pointed out [97], aggregate fitnesses strongly reduce the information conveyed to GP about i relative to the description of $i's$ behavior available in $e(i)$, thereby under-utilizing information that could help guide the search. In addition, many forms of aggregation assume all tests are equally informative (although there are exceptions, including implicit fitness sharing which is discussed below). Therefore individuals that are elite (i.e. have the lowest error in the population) for portions of e (i.e., locally) are not selected if they perform poorly in other regions and therefore have a higher F. By providing equivalent selection pressure with respect to test cases, GP misses the opportunity to identify and/or preserve programs that perform especially well in certain regions of the problem, most importantly those portions of the problem that are more difficult for the process to solve. We expect GP to solve problems through the induction, propagation and recombination of

building blocks (i.e. subprograms) that provide partial solutions to our desired task. Hence we wish to select those programs that imply a partial solution by performing uniquely well on subsets of the problem.

Several methods have been proposed to reward individuals with uniquely good test performance, such as implicit fitness sharing (IFS) [134], historically assessed hardness [89], and co-solvability [96], all of which assign greater weight to fitness cases that are judged to be more difficult in view of the population performance. Perhaps the most effective parent selection method recently proposed is lexicase selection [60, 193]. In particular, "global pool, uniform random sequence, elitist lexicase selection" [193], which we refer to simply as lexicase selection, has outperformed other similarly-motivated methods in recent studies [61, 118]. Despite these gains, it fails to produce such benefits when applied to continuous symbolic regression problems, due to its method of selecting individuals based on test case elitism. We demonstrated recently [109] that by re-defining the test case pass condition in lexicase selection using an adaptive ϵ threshold, the benefits of lexicase selection can be achieved in continuous domains. I show in this chapter that by combining ϵ-lexicase selection with local model seeding, a general, reduced-order model of VIV can be constructed that accurately represents the changing response of a cylindrical structure to varying fluid forces.

§8.3 describes the VIV problem that forms the experimental basis of our study of generalization methods. In §8.4, I describe ϵ-lexicase selection and the seeding process. Related work is summarized in §8.5. The experimental analysis in §8.6 describes how the algorithm is implemented for VIV, and the results of this analysis are presented in §8.7, including local and global models. The modeling results are discussed in §8.8, including recommendations for future study.

8.3 Vortex Induced Vibration

VIV is a fluid-structure interaction (FSI) phenomenon that occurs in structures subject to fluid disturbances, such as bridges, chimney stacks, and tethered offshore structures such as the mooring lines that secure floating wind turbines [7, 13, 153, 154, 155, 174]. A review

of experimental research in this area is presented in [218]. Although the dynamics of VIV can be captured by solving the Navier-Stokes equations, the computational intensity of that approach, as well as the complexity of the underlying models, motivates the development of accurate, reduced-order models.

In §2.6.2.1, data from a single flow velocity was used to update a first principles model of the system proposed by Faccinetti [41] using MSAM. Although such adaptations can improve model fidelity locally, in general the interaction between the flow force, q, and the transverse cylinder motion x varies considerably as the crossflow velocity of the fluid U changes, making global modeling difficult. During synchronization, known as lock-in, the frequency of vortex shedding, oscillation of the structure normal to the flow, and the structure's natural frequency in the fluid, f_n, coincide, resulting in lightly-damped, high-amplitude oscillations of the structure normal to the flow direction.

The experiment studied here was carried out by the UMass Fluid Structure Interaction Laboratory. Data was collected for a 14.6 mm diameter cylinder of 30 cm length, placed in a re-circulating water tunnel, with a test section of 1.27 m $\times$ 0.5 m $\times$ 0.38 m, a turbulence intensity of less than 1% for up to a flow velocity of $U = 0.3$ m/s and a velocity uniformity of less than 2% [185, 184]. The set-up used to hold the cylinder in the test section had two air bearings to reduce the damping and constrain the oscillations of the cylinder to one degree of freedom in the crossflow direction. Springs were attached from the supporting plate holding the cylinder to the fixed housing.

The data are presented as a function of the reduced flow velocity, $U^* = \frac{U}{f_n D}$, where D is the cylinder diameter. The cylinder's displacement and the corresponding flow forces were simultaneously measured at 19 constant reduced flow velocities ranging from $U^* = 4.4$ to $U^* = 10.9$ at 500 Hz for 50 seconds at each velocity. The varying interaction between q and x is notable in the phase $\angle(q, x)$ at f_n, shown in Figure 8.1 for different flow velocities. At flow velocities below lock-in, the cylinder oscillates in phase with the flow forces; during lock-in, which begins at $U^* \approx 6$, the angle between q and x shifts from 0° to perpendicular to the

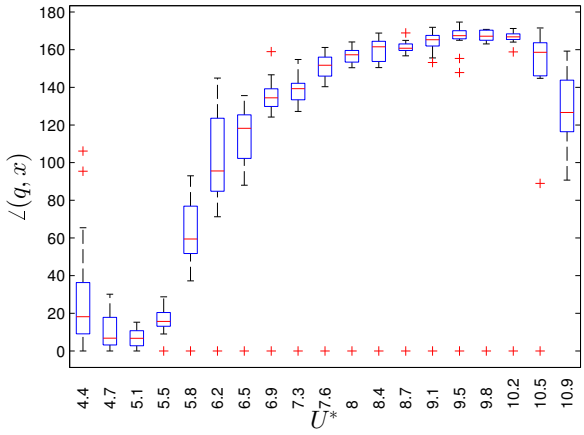

Figure 8.1. Phase between flow force and cylinder displacement for different true reduced velocities U^*.

flow forces, and as the flow velocity continues to increase, x and q become completely out of phase.

As discussed in Ch. 2, Faccinetti [40] proposed to model the limit cycle characteristics of this process as a mass spring damper excited by a van der Pol oscillator, i.e.,

$$(m_s + 1/4\pi C_M \rho D^2)\ddot{x} + [r_s 2\pi \mathrm{St}(U/D)\rho D^2]\dot{x} + hx = 1/4\rho U^2 D C_{Lo} q \tag{8.2}$$

$$\ddot{q} + \epsilon[2\pi \mathrm{St}(U/D)](q^2 - 1)\dot{q} + [2\pi \mathrm{St}(U/D)]^2 q = (A/D)\ddot{x} \tag{8.3}$$

where St is the Strouhal number, m_s is the mass of the structure, ρ is the fluid density, r_s represents viscous dissipation in the support, γ denotes the stall parameter, C_M is the added mass coefficient, C_{Lo} is the lift coefficient, and ϵ and A are the van der Pol scaling parameters. When compared to the experimental data, Eq. 8.3 fails to accurately model the process, as shown by the mismatch between the left and ride hand sides for different values of U^* in Figure 8.2.

We wish to identify a 2nd-order model of the cylinder displacement and the fluid force that produces accurate estimates across flow velocities. Given that the test specimen is a rigid cylinder constrained to behave as a mass spring damper, we model the cylinder state

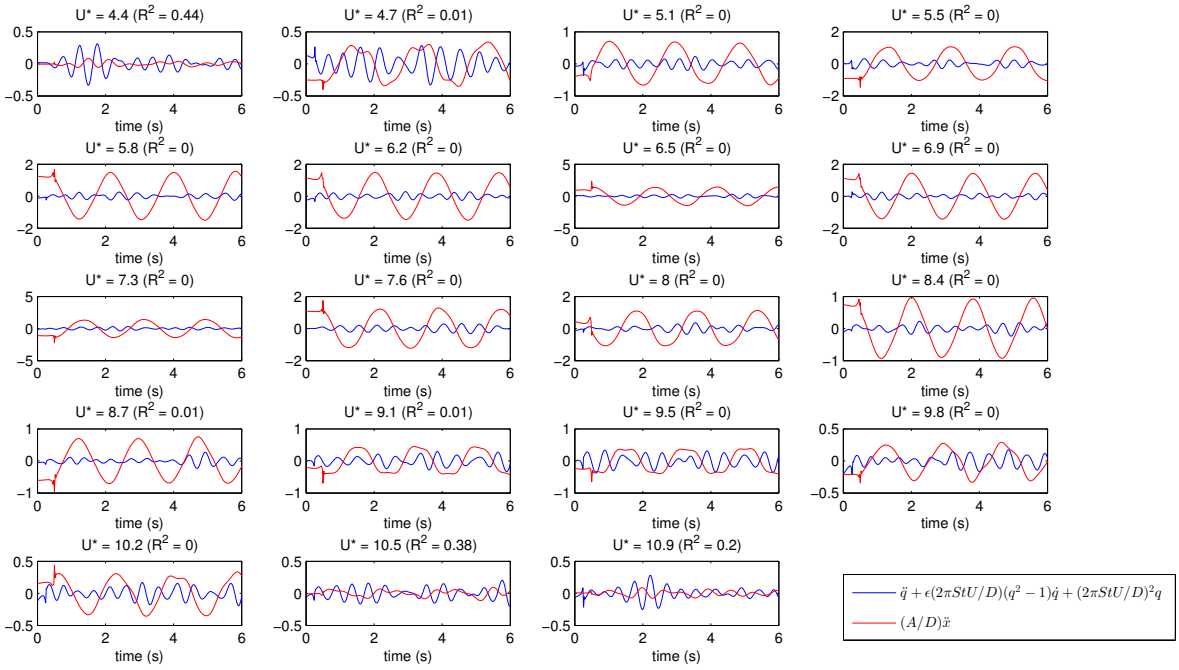

Figure 8.2. Comparison of the left and right-hand sides of Eq. 8.3 for different true reduced velocities U^*.

as a mass spring damper as in Eq. 8.2, and limit identification to the parameters associated with each state, as

$$\ddot{x} = \theta_1 \dot{x} + \theta_2 x + \theta_3 q \tag{8.4}$$

The parameters θ_1, θ_2, and θ_3 are estimated in simulation using non-linear least squares [78]. The model structure search is thus constrained to identification of a fluid force model as in Eq. (8.3). Using ELGP, we attempt to identify the fluid force dynamics as

$$\ddot{q} = f(\dot{q}, q, \ddot{x}, \dot{x}, x)$$

For the purposes of identification, $\ddot{q}$, $\dot{q}$, $\ddot{x}$ and $\dot{x}$ are estimated numerically, as described in Ch. 3. This allows model forms to be evaluated numerically.

8.4 ϵ-lexicase selection

Lexicase selection is a parent selection technique based on lexicographic ordering of test (i.e. fitness) cases. Each parent selection event proceeds as follows:

1. The entire population is added to the selection pool.
2. The fitness cases are shuffled.
3. Individuals in the pool with a fitness worse than the best fitness on this case among the pool are removed.
4. If more than one individual remains in the pool, the first case is removed and 3 is repeated with the next case. If only one individual remains, it is the chosen parent. If no more fitness cases are left, a parent is chosen randomly from the remaining individuals.

As evidenced above, the algorithm is quite simple to implement. In this procedure, test cases act as filters, and a randomized path through these filters is constructed each time a parent is selected. Each parent selection event returns a parent that is elite on at least the first test case used to select it. In turn, the filtering capacity of a test case is directly proportional to its difficulty since it culls the individuals from the pool that do not do the best on it. Therefore selective pressure continually shifts to individuals that are elite on cases that are not widely solved in the population. Because each parent is selected via a randomized ordering of test cases and these cases perform filtering proportional to their difficulty, individuals are pressured to perform well on unique combinations of test cases, which promotes individuals with diverse performance, leading to increased diversity observed during evolutionary runs [60].

Lexicase selection was originally applied to multimodal [193] and "uncompromising" [60] problems. An uncompromising problem is one in which only exact solutions to every test case produce a satisfactory program. For those types of problems, using each case as a way to select only elite individuals is well-motivated, since each test case must be solved exactly. In regression, exact solutions to test cases can only be expected for synthetic problems, whereas real-world problems are subject to noise and measurement error. With respect to the lexicase

selection process, continuously-valued errors are problematic, due to the fact that individuals in the population are not likely to share elitism on any particular case unless they are identical equations. On regression problems, the standard lexicase procedure typically uses only one case for each parent selection, resulting in poor performance.

We hypothesize that lexicase selection performs poorly on continuous errors because the case passing criteria is too stringent in continuous error spaces. For individual i to pass case t, lexicase requires that $e_t(i) = e_t^*$, where e_t^* is the best error on that test case in the pool. To remedy this shortcoming, we introduced ϵ-lexicase selection [109], which modulates the pass condition on test cases via a parameter ϵ, such that only individuals outside of a predefined ϵ are filtered in step 3 of lexicase selection. We found that it is best to automatically adapt the ϵ threshold to take into account the values of $e_t(i)$ across P, denoted $\mathbf{e}_t \in \mathbb{R}^{|P|}$, so that it can modulate its selectivity based on the difficulty of case t. A common estimate of difficulty in performance on a fitness case is variance [175]; in this regard ϵ could be defined according to the standard deviation of $\mathbf{e}_t$, i.e. $\sigma(\mathbf{e}_t)$. Given the high sensitivity of σ to outliers, however, we opt for a more robust estimation of variability by using the median absolute deviation (MAD) [157] of $\mathbf{e}_t$, defined as

$$MAD(\mathbf{e}_t) = \lambda(\mathbf{e}_t) = \text{median}_j \left(|\mathbf{e}_{t_j} - \text{median}_k(\mathbf{e}_{t_k})| \right) \tag{8.5}$$

We use Eq. (8.5) in the definition of two ϵ values, ϵ_e and ϵ_y, that control that pass condition $p_t(i)$ as

$$\epsilon_e : p_t(i) = \mathbb{I}\left(e_t(i) < e_t^* + \lambda(\mathbf{e}_t)\right) \tag{8.6}$$

$$\epsilon_y : p_t(i) = \mathbb{I}\left(e_t(i) < \lambda(\mathbf{e}_t)\right) \tag{8.7}$$

Here $\mathbb{I}$ is the indicator function that returns 1 if true and 0 if false. As shown in Eq. (8.6), ϵ_e defines $p_t(i)$ relative to e_t^*, and therefore is always passed by at least one individual in P. Conversely, ϵ_y (Eq. (8.7)) defines $p_t(i)$ relative to the target value y_t, meaning that $\hat{y}_t(i)$ must

be within $\pm\epsilon_y$ of y_t to pass case t. In this way ϵ_y provides no selection pressure if there is not an individual in the population within adequate range of the true value for that case.

An important consideration in parent selection is the time complexity of the selection procedure. Lexicase selection has a theoretical worst-case time complexity of $O(|P|^2 N)$, compared to a time complexity of $O(|P|N)$ for tournament selection. Although clearly undesirable, this worst-case complexity is only reached if every individual passes every test case during selection; in practice [60], lexicase selection normally uses a small number of cases for each selection and therefore incurs only a small amount of overhead. The wall clock times for our variants of lexicase compared to other methods were quantified in [109] and showed negligible differences.

8.5 Related Work

Although to an extent the ideas of multiobjective optimization apply to multiple test cases, they are qualitatively different: objectives are the defined goals of a task, whereas test cases are tools for estimating progress towards those objectives. Objectives and test cases therefore commonly exist at different scales: symbolic regression often involves one or two objectives (e.g. accuracy and model conciseness) and hundreds or thousands of test cases. One example of using test cases explicitly as objectives occurs in Langdon's work on data structures [113] in which small numbers of test cases (in this case 6) are used as multiple objectives in a Pareto selection scheme. Other multi-objective approaches such as NSGA-II [32], SPEA2 [220] and ParetoGP [190] are used commonly with a small set of objectives in symbolic regression. The "curse of dimensionality" prevents the use of objectives at the scale of typical test case sizes, since most individuals become nondominated[1], leading to selection based mostly on expensive diversity measures rather than performance. Scaling issues in many-objective optimization are reviewed in [76]. In lexicase selection, parents are guaranteed to be nondominated with respect to the fitness cases. Pareto strength in SPEA2

[1]Program i_1 dominates i_2 if $f_j(i_1) \leq f_j(i_2)\ \forall j$ and $f_j(i_1) < f_j(i_2)$ for at least one j (f is minimized).

promotes individuals based on how many individuals they dominate, and similarly lexicase selection increases the probability of selection for individuals who solve *more* cases and *harder* cases (i.e. cases that are not solved by other individuals) and decreases for individuals who solve *fewer* or *easier* cases.

A number of GP methods attempt to affect selection by weighting test cases based on population performance. In non-binary Implicit Fitness Sharing (IFS) [98], the fitness proportion of a case is scaled by the performance of other individuals on that case. Similarly, historically assessed hardness scales error on each test case by the success rate of the population [89]. Discovery of objectives by clustering (DOC) [97] clusters test cases by population performance, and thereby reduces test cases into a set of objectives for search. Both IFS and DOC were outperformed by lexicase selection on program synthesis and boolean problems in previous studies [61, 118]. Other methods attempt to sample a subset of $\mathcal{T}$ to reduce computation time or improve performance, such as dynamic subset selection [48], interleaved sampling [51], and co-evolved fitness predictors [175]. Unlike these methods, lexicase selection begins each selection with the full set of training cases, and allows selection to adapt to program performance on them.

The conversion of a model's real-valued fitness into discrete values based on an ϵ threshold has been explored in other research; for example, Novelty Search GP [130] uses a reduced error vector to define behavioral representation of individuals in the population. This paper proposes it for the first time as a solution to applying lexicase selection effectively to regression.

As a behavioral-based search driver, lexicase selection belongs to a class of GP systems that attempt to incorporate a program's behavior explicitly into the search process, and as such shares a general motivation with recently proposed methods such as Semantic GP [139] and Behavioral GP [99], despite differing strongly in approach. Although lexicase is designed with behavioral diversity in mind, recent studies suggest that structural diversity can also significantly affect GP performance [20].

Table 8.1. Global modeling settings.

Setting	Value
Population size	1000
Crossover / mutation	80/20%
Program length limits	[3, 20]
ERC range	[-1,1]
Generation limit	1000
Trials	30
Terminal Set	{**x**, ERC, +, −, ∗, /, sin, cos, exp, log}
Elitism	keep best

8.6 Experimental Analysis

Since the dynamics were not observed to exceed 5 Hz, the data was filtered to 10 Hz for identification. Each set was split 50/50 into training and test sets. Identification was first performed on each data set collected at specific flow velocities. 10 trials of age-fitness Pareto optimization (AFP, see §3.6.1) were used to train the models. During identification, solutions are archived based on Pareto-dominance in complexity and fitness (as in Ch. 6). The solution archives are then consolidated into a seed population, and global identification is performed on the cascaded training sets.

Global identification of the fluid force dynamics is performed on the cascaded data sets, preserving the training and test partitions. To assess the effect of seeding the initial population and using ϵ-lexicase selection, 30 trials of six treatments were tested, with '+S' indicating a seeded initial population: AFP, AFP+S, Lex ϵ_y, Lex ϵ_y+S, Lex ϵ_e, and Lex ϵ_e+S. The ELGP settings are shown in Table 8.1.

8.7 Results

The local model identification is presented first, in terms of the best fit models as well as the archives that produce the seeded solutions for global identification. Then the global models are presented. We compare the six treatments according to the fitness of the models they produce. Finally, the best fit model is evaluated and simulated along with the parameterized mass spring damper model. The modeling results are compared to measurement data for each flow velocity.

Table 8.2. Best fit local models..

U*	Equation	Complexity	Train R^2	Test R^2
4.4	$\ddot{q} = 18.8\,x - 2.08\,q + 18.8\,\ddot{x}$	28	1.00	0.92
4.7	$\ddot{q} = x - 7\,q - \dot{x}\ (x\,\dot{x} - 0.186)$	15	0.90	0.91
5.1	$\ddot{q} = x - 1.14\,q + 0.138\,\dot{x}$	9	0.82	0.88
5.5	$\ddot{q} = 3.28\,x - q + 2.93\,\ddot{x} + 0.155\,\dot{x}$	15	0.92	0.87
5.8	$\ddot{q} = 0.177\,\dot{x} - 0.0351\,\ddot{x} - 0.955\,q$	13	0.85	0.81
6.2	$\ddot{q} = 0.0446\,\ddot{x} - 0.995\,q + 0.172\,\dot{x}$	11	0.80	0.85
6.5	$\ddot{q} = 0.18\,\dot{x} - 1.11\,q - 0.18\,\log(\ddot{x})\ (x + 0.493)$	22	0.87	0.85
6.9	$\ddot{q} = 0.156\,\ddot{x} - q + 0.156\,\dot{x}$	7	0.85	0.82
7.3	$\ddot{q} = 0.169\,\ddot{x} - q + 0.169\,\dot{x} - 0.169\,\log(\ddot{x}\,\dot{x})$	24	0.87	0.83
7.6	$\ddot{q} = 0.693\,x - q + \ddot{x} + 0.146\,\dot{x}$	11	0.95	0.89
8.0	$\ddot{q} = 0.35\,\ddot{x} - q + 0.183\,\dot{x}$	9	0.92	0.92
8.4	$\ddot{q} = 0.53\,\ddot{x} - 1.16\,q + 0.159\,e^{\dot{x}} - 0.25$	20	0.88	0.86
8.7	$\ddot{q} = 0.606\,\ddot{x} - 1.19\,q + 0.206\,\dot{x}$	11	0.90	0.88
9.1	$\ddot{q} = \ddot{x}\ (0.0755\,q + \ddot{x} + 0.612) - q - 0.278$	13	0.95	0.95
9.5	$\ddot{q} = \frac{1.17\,(x - 0.21)}{\log(\dot{x}) - 0.0893} - 1.17\,q - 0.237$	68	0.94	0.86
9.8	$\ddot{q} = x - 1.35\,q + 3.71\,\ddot{x} + 1.35\,\sin(1.34\,x)$	24	0.92	0.91
10.2	$\ddot{q} = 0.871\,\ddot{x} - 1.1\,q$	7	0.89	0.93
10.5	$\ddot{q} = -2.07\,q - 0.346\,x - 2.07\ (x + \ddot{x})\ (\ddot{x} - 8.96)$	15	0.99	0.91
10.9	$\ddot{q} = 2.08\,U^2\ (x + \ddot{x} + 0.0164\,\dot{x}) - 2.08\,q - 0.00754$	34	0.97	0.96

8.7.1 Local Models

The best local models for each data set are listed in Table 8.2. The correlations on the test set range from 0.81 to 0.96 depending on the data set. Figure 8.3 shows the archives resulting from local identification. The models marked by red squares represent the Pareto front of the archived results of the trials. These models are inserted into the initial population of the global identification runs.

8.7.2 Global Models

The best-of-run model R^2 values on the test data are shown in Figure 8.4 and demonstrate the improved model quality that results from 1) using ϵ-lexicase selection and 2) seeding the initial populations with locally fit models. The ϵ-lexicase selection treatments all produce models with better median R^2 values than AFP and AFP+S; in addition, the seeding for Lex ϵ_y and Lex ϵ_e generates a significant improvement in the median model quality. The head start in model quality produced by seeding is also evident in the progress of best fit models during the algorithms' executions, as shown in Figure 8.5. The seeded runs start with

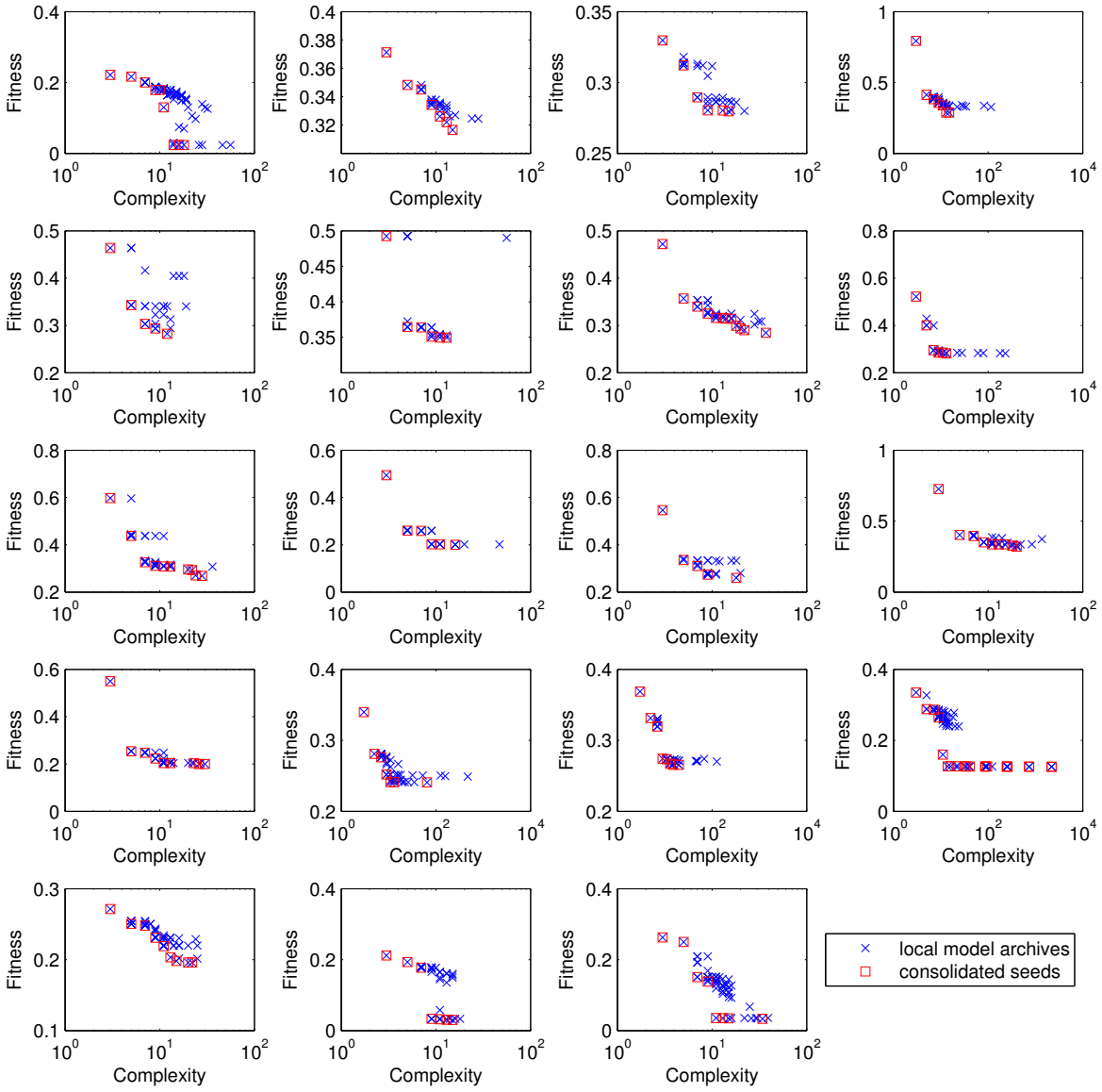

Figure 8.3. ELGP runs on local data sets generate the models marked with x. These are consolidated into a single archive based on Pareto dominance to seed the initial population of ELGP trained on the global data set.

$\approx$ 20% lower MAE values. In addition, the lexicase treatments converge much quicker on fitter solutions than the AFP treatments.

A consolidated Pareto archive of all of the modeling results is shown in Figure 8.6. In this case, complexity is plotted on the y-axis, and fitness is plotted on the x-axis, to allow the model forms to be shown. The fluid force model that achieves the best test fitness has the following form:

$$\ddot{q} = \sin\left(\theta_4 \ddot{x} + \theta_5 x - q + \theta_6 \dot{x}\right) - q \tag{8.8}$$

with θ_4 = 18.009, θ_5 = 17.827, and θ_6 0.341. Under algebraic evaluation this model has an average correlation of R^2 = 0.94 with $\ddot{q}$ on the test data. The parameters of the cylinder displacement (Eq. (8.4)) are estimated via nonlinear least squares as θ_1 = -11.9411, θ_2 = -0.1955, and θ_3 = 2.0527. The cylinder displacement shows good agreement with measurement data as expected, with an average correlation of R^2 = 0.96.

To evaluate Eq. (8.8) in its prediction of q, we simulate its behavior using initial conditions $(q(0), \dot{q}(0))$ chosen from the test sets at different U^*. The results for six seconds of simulation at each flow velocity are compared to measurements in Figure 8.7. The results indicate good agreement with the measurements. However, we note that for simulation lengths longer than 10 seconds, the predictions begin to diverge from measured values, indicating that the model may only be locally stable.

Eqns. (8.4) and (8.8) are simulated in a decoupled fashion, meaning that the states are integrated only with respect to their own derivatives. Under these conditions, accurate phase portraits of the VIV dynamics can be generated at each flow velocity, as shown in Figure 8.8.

8.8 Discussion and Conclusion

The results demonstrate that ϵ-lexicase selection with seeding provides a viable approach to generalization for modeling dynamic systems with GP. In comparison to the AFP algorithm used for ELGP, ϵ-lexicase selection is shown to converge more quickly and produce higher

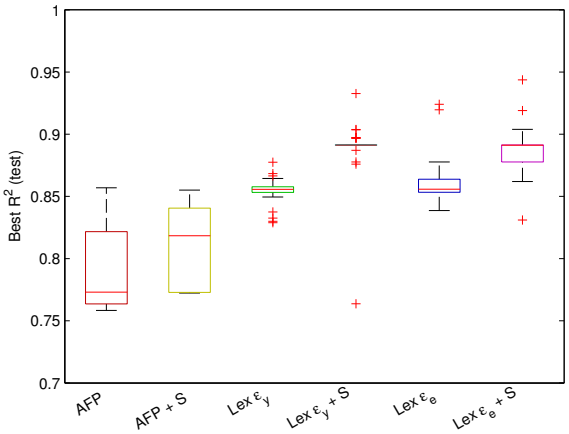

Figure 8.4. R^2 values of best models from 30 trials of the different methods. '+ S' indicates that the method was seeded with local model archives.

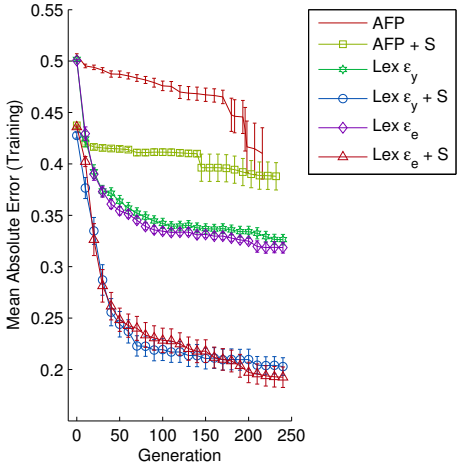

Figure 8.5. Mean absolute error convergence on the training data using the different methods. The seeded cases begin with lower fitness models, as expected.

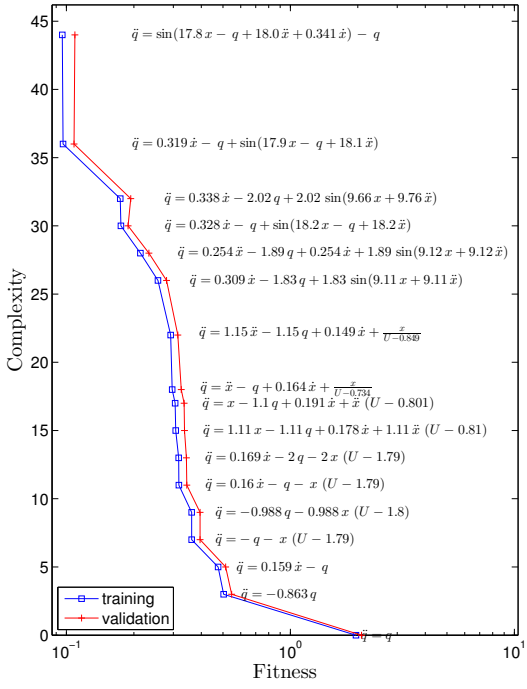

Figure 8.6. Consolidated Pareto archive of global models.

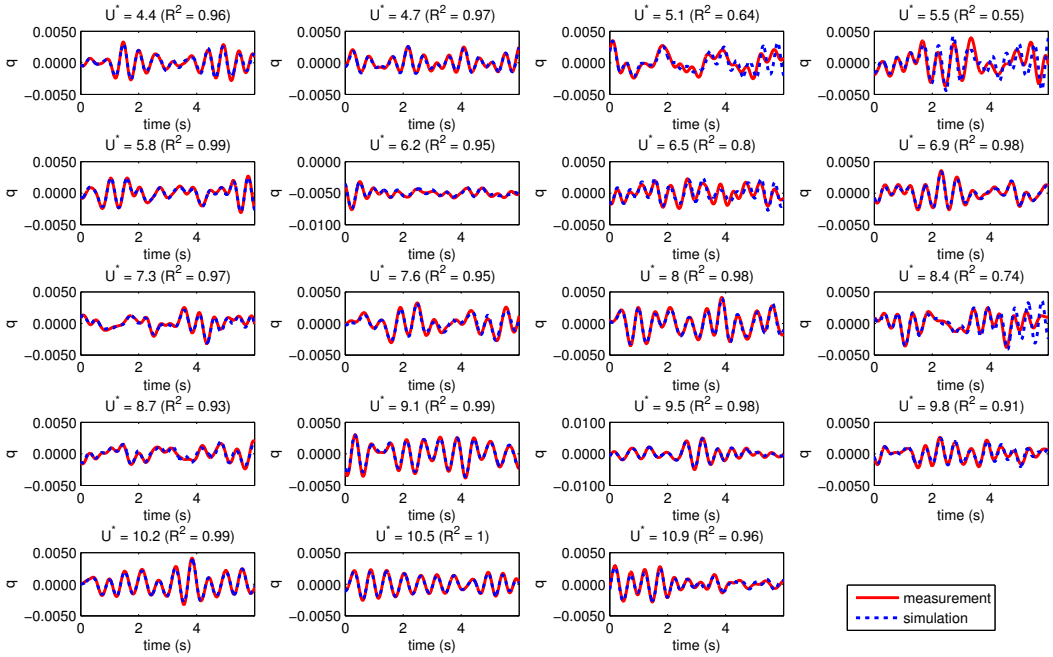

Figure 8.7. Model of q compared to measurement data for different flow velocities.

fidelity models. We are able to create a reduced-order model of the VIV phenomenon that captures the cylinder displacement and fluid force as they vary with U^*. The final model is of the form

$$\ddot{x} = \theta_1 \dot{x} + \theta_2 x + \theta_3 q$$

$$\ddot{q} = \sin\left(\theta_4 \ddot{x} + \theta_5 x - q + \theta_6 \dot{x}\right) - q$$

Challenges to fully capturing the VIV dynamics remain. In particular, a fully coupled simulation that remains stable for more than 10 seconds has yet to be accomplished. Although we have shown that the models produce accurate results given appropriate measured inputs, the fully coupled simulation will allow researchers to begin to explore more aspects of VIV beyond the behavior modeled here. In addition, it is of great interest to apply this modeling tool to different experimental setups such that the generalizability of the identified model with respect to system parameters can be analyzed.

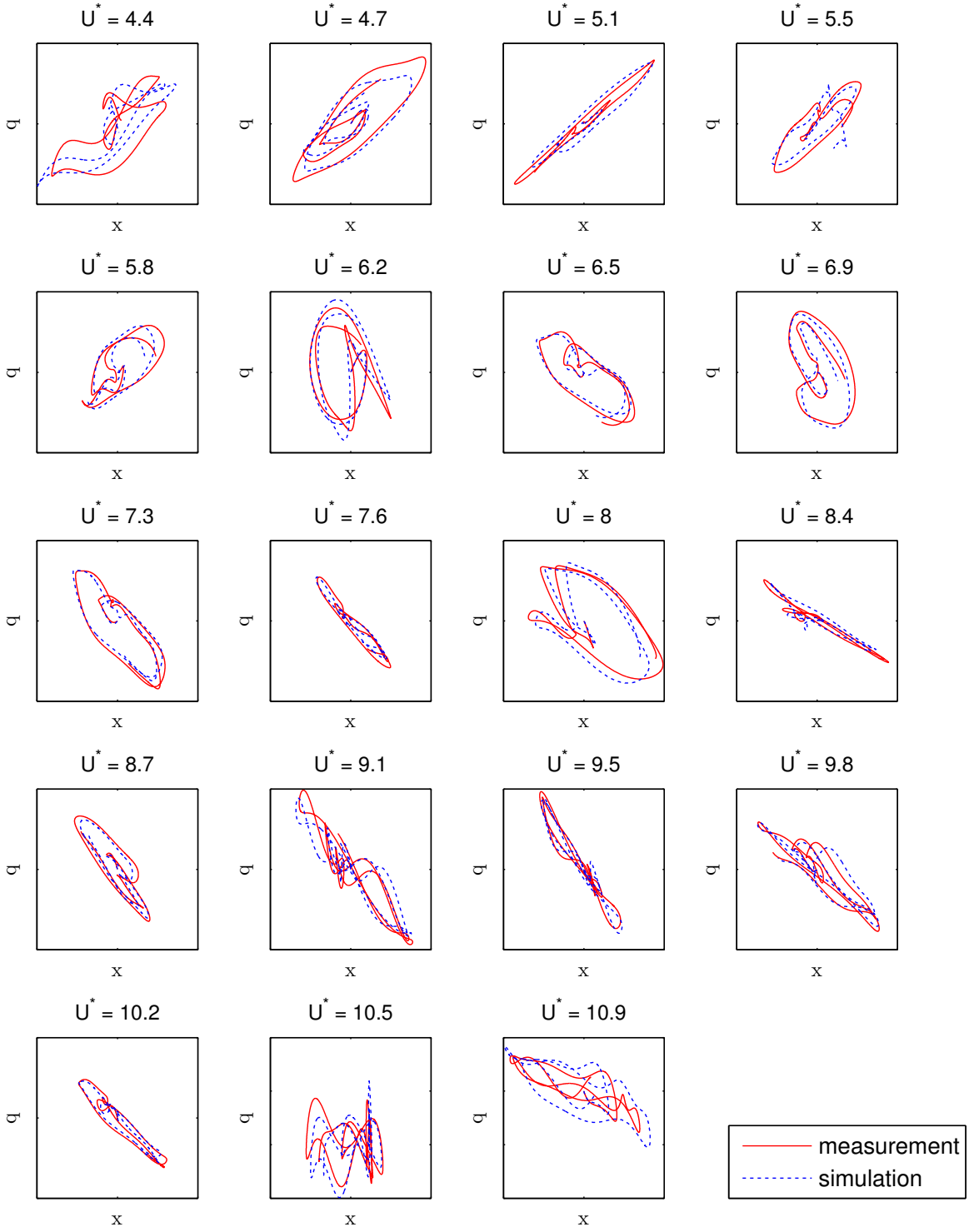

Figure 8.8. Simulated and measured phase portaits of q and x for different flow velocities.

8.9 Acknowledgments

I would like to thank Yahyah Modarres-Sadeghi and Banafsheh Seyed-Aghazadeh for providing the VIV data and help with the analysis, and Thomas Helmuth, Nic McPhee and Bill Tozier for their feedback on ϵ-lexicase selection.

CHAPTER 9

CONCLUSION

This dissertation presents three methods that are designed to improve the capacity of system identification methods to identify succinct and accurate model structures. Model structures that are intelligible are more adept at informing experts of their embedded knowledge. The challenge of succinct nonlinear modeling pervades disciplines across the spectrum of science and engineering; here, it has been addressed for three scenarios. The first scenario applies to continuous dynamic processes for which experts have designed intelligible models or controllers that fail to fully explain or control the nonlinearities of the process exhibited in measured observations. MSAM is designed to optimize the introduction of nonlinear couplings to these models that improve their performance while maintaining their intelligibility. The second scenario applies to continuous dynamic processes for which no accurate starting models are available. ELGP is designed to produce concise model structures achieved by improving the capacity and search for such models in GP. The third scenario applies to multiclass dynamic processes, like the behavior of bald eagles, for which no starting model is available. M4GP is designed to produce models for this task by using GP as a feature engine that can perform feature selection as well as feature synthesis, resulting in succinct models represented as transformations of the original feature space. These methods are demonstrated through application to the identification of nonlinear dynamics for control design, wind turbine modeling, bald eagle behavioral modeling, and fluid-structure interaction modeling.

Together, MSAM, ELGP, and M4GP address several of the challenges to succinct nonlinear model structure identification. The rest of this chapter is devoted to identifying the challenges they do *not* address, and to providing some insights into future research directions.

Whereas the discussion sections in the respective chapters identify some of the method-specific extensions, the goal of this discussion is to identify more broadly the challenges to model structure identification for dynamic processes. The first research question discussed in §9.1 is the trade-off between algebraic and simulation-based evaluation of candidate models, which deserves a more thorough treatment. The second research question stems from the insight that MSAM, ELGP and M4GP are hybrid methods. §9.2 discusses what role deterministic and stochastic machine learning methods could play in the future of system identification.

9.1 Model Evaluation

As mentioned in Chs. 2 and 3, the common choice for estimating the model output(s) is numerical integration (i.e., simulation) of state variables, i.e. the "output error" method [121]. However, given the sensitivity of simulation to different model structures and the computational cost of numerical integration, the alternative approach of algebraically estimating candidate model outputs is preferred for symbolic regression [15, 177]. In the algebraic approach, un-measured states, denoted $\tilde{\mathbf{x}}$, are estimated from measurements via numerical differentiation together with smoothing functions. This yields an algebraic estimate for the prediction error, given in Eq. 3.2.

The algebraic prediction error will differ from the error yielded by simulated-based evaluation. To illustrate why, consider the classical Runge-Kutta method (RK4) to solving an ordinary differential equation:

$$\dot{y} = f(t, y) \ , \ y(t_0) = y_0$$

then for step size h,

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2(k_2 + k_3) + k_4) \ , \ t_{n+1} = t_n + h$$

where

$$
\begin{aligned}
k_1 &= f(t_n, y_n) \\
k_2 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \\
k_3 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2) \\
k_4 &= f(t_n + h, y_n + hk_3)
\end{aligned}
$$

now consider a model $\hat{\dot{y}} = \hat{f}(t, y)$ with algebraic error $\epsilon = \dot{y} - \hat{\dot{y}}$, such that $\hat{\dot{y}} = \dot{y} - \epsilon = f(t, y) - \epsilon(t)$. Then the error propagates through simulation as

$$
\begin{aligned}
\hat{y}_n &= y_n - \epsilon_n \\
\hat{k}_1 &= \hat{f}(t_n, y_n - \epsilon_n) \\
\hat{k}_2 &= \hat{f}(t_n + \frac{h}{2}, y_n - \epsilon_n + \frac{h}{2}\hat{f}(t_n, y_n - \epsilon_n)) \\
\hat{k}_3 &= \hat{f}(t_n + \frac{h}{2}, y_n - \epsilon_n + \hat{f}(t_n + \frac{h}{2}, y_n - \epsilon_n + \frac{h}{2}\hat{f}(t_n, y_n - \epsilon_n))) \\
\hat{k}_4 &= \hat{f}(t_n + h, y_n - \epsilon_n + \\
&\quad h\hat{f}(t_n + \frac{h}{2}, y_n - \epsilon_n + \hat{f}(t_n + \frac{h}{2}, y_n - \epsilon_n + \frac{h}{2}\hat{f}(t_n, y_n - \epsilon_n))))
\end{aligned}
$$

Thus the error at a particular time step is amplified in the estimation of subsequent time steps. The algebraic evaluation can be considered optimistic in the sense that it assumes the state variables $\mathbf{x}$ to be free of error propagated from previous time steps. The argument for this approach is that the algebraic error more accurately reflects the differences between competing model structures since it is not masked by error propagation through simulation. It also requires only one evaluation of each model on the target data, rather than 4, in the case of RK4. However, problems can arise when simulating candidate models that have been chosen via algebraic evaluation, because their robustness to error propagation has not been quantified. An example of this is noted in Ch. 8, where identified models drifted from measurement after many time steps.

Future work should address the difference between algebraic and simulated-based error more analytically to justify the use of one approach, the other, or both. An interesting topic could be to investigate algebraic estimates of simulation-based error propagation to improve algebraic estimates of simulation error without having to resort to expensive simulations during identification. Identification could also proceed through stages in which certain models in the population are chosen for simulation to enhance confidence in their capacity for generalization.

9.2 Hybrid Methods

The three methods presented in this dissertation represent hybrid approaches to model structure identification. MSAM intertwines exhaustive search with parameter estimation; ELGP interwines stochastic hill climbing with GP; M4GP intertwines GP with distance-based classification. These methods are part of a growing body of research that combines stochastic model structure identification algorithms with deterministic algorithms to achieve a balance between structural and parametric search [201, 132, 72, 99, 2, 3]. The abundance of disparate hybrid algorithms suggests that a generalized theory for interfacing the strengths of GP methods (e.g. feature creation and selection, model structure optimization) with the strengths of fast machine learning algorithms (e.g. ordinary least squares, decision trees, naïve Bayes, etc.) has yet to be realized. Recent methods like Behavioral GP [99] treat the subprogram outputs of GP individuals as features for data mining, and the data mining in turn determines which sub-components are important to share. At the other end of the spectrum, several methods treat individual model outputs as features in a single ensemble model [85, 132, 72, 31, 3]. In other words, the balance and scale of GP in relation to other methods has not been decided. Future work could explore the trade-offs and the scales at which hybrid GP algorithms are most effective.

BIBLIOGRAPHY

[1] Aben, Job, Strubbe, Diederik, Adriaensen, Frank, Palmer, Stephen C. F., Travis, Justin M. J., Lens, Luc, and Matthysen, Erik. Simple individual-based models effectively represent Afrotropical forest bird movement in complex landscapes. *Journal of Applied Ecology 51*, 3 (June 2014), 693–702.

[2] Arnaldo, Ignacio, Krawiec, Krzysztof, and O'Reilly, Una-May. Multiple regression genetic programming. In *Proceedings of the 2014 conference on Genetic and evolutionary computation* (2014), ACM Press, pp. 879–886.

[3] Arnaldo, Ignacio, O'Reilly, Una-May, and Veeramachaneni, Kalyan. Building Predictive Models via Feature Synthesis. ACM Press, pp. 983–990.

[4] Aström, Karl J., and Wittenmark, Björn. *Adaptive Control*, second ed. Dover Publications, Mineola, NY USA, 2008.

[5] Banzhaf, Wolfgang. Genotype-phenotype-mapping and neutral variation - a case study in genetic programming. In *Parallel problem solving from nature (PPSN) III*. Springer, 1994, pp. 322–332.

[6] Bates, Douglas M, and Watts, Donald G. *Nonlinear regression analysis and its applications*. John Wiles & Sons, Inc, New York, NY, USA, 1988.

[7] Bearman, Peter W. Vortex shedding from oscillating bluff bodies. *Annual Review of Fluid Mechanics 16*, 1 (1984), 195–222.

[8] Belding, Theodore C. The Distributed Genetic Algorithm Revisited. *arXiv:adap-org/9504007* (May 1995). arXiv: adap-org/9504007.

[9] Bianchi, F. D., Sánchez-Peña, R. S., and Guadayol, M. Gain scheduled control based on high fidelity local wind turbine models. *Renewable Energy 37*, 1 (Jan. 2012), 233–240.

[10] Billings, S. A. *Nonlinear System Identification*, 1st ed. Wiley, Chichester, West Sussex, UK, 2013.

[11] Billings, S. A., and Zhu, Q. M. Nonlinear model validation using correlation tests. *International Journal of Control 60*, 6 (1994), 1107–1120.

[12] Billings, Stephen A. *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[13] Blevins, Robert D. *Flow-induced vibration*. Krieger Pub. Co., Malabar, Fla., 1990.

[14] Bongard, J.C., and Lipson, H. Nonlinear System Identification Using Coevolution of Models and Tests. *IEEE Transactions on Evolutionary Computation 9*, 4 (Aug. 2005), 361–384.

[15] Bongard, Josh, and Lipson, Hod. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences 104*, 24 (2007), 9943–9948.

[16] Bossanyi, E. A. Wind Turbine Control for Load Reduction. *Wind Energy 3*, 3 (2000), 149–163.

[17] Bououden, S., Chadli, M., Filali, S., and El Hajjaji, A. Fuzzy model based multivariable predictive control of a variable speed wind turbine: LMI approach. *Renewable Energy 37*, 1 (Jan. 2012), 434–439.

[18] Brameier, Markus, and Banzhaf, Wolfgang. *Linear Genetic Programming*, 1 ed., vol. 1. Springer, 2007.

[19] Breiman, Leo. Random forests. *Machine learning 45*, 1 (2001), 5–32.

[20] Burks, Armand R., and Punch, William F. An Efficient Structural Diversity Technique for Genetic Programming. ACM Press, pp. 991–998.

[21] Cao, Hongqing, Kang, Lishan, Chen, Yuping, and Yu, Jingxian. Evolutionary modeling of systems of ordinary differential equations with genetic programming. *Genetic Programming and Evolvable Machines 1*, 4 (2000), 309–337.

[22] Caruana, Rich, and Niculescu-Mizil, Alexandru. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning* (2006), ACM, pp. 161–168.

[23] Chavarriaga, Ricardo, Sagha, Hesam, Calatroni, Alberto, Digumarti, Sundara Tejaswi, Trster, Gerhard, Milln, Jos del R., and Roggen, Daniel. The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters 34*, 15 (2013), 2033–2042.

[24] Chen, Sheng, Billings, Stephen A., and Luo, Wan. Orthogonal least squares methods and their application to non-linear system identification. *International Journal of control 50*, 5 (1989), 1873–1896.

[25] Chipperfield, Andrew, and Fleming, Peter. Multiobjective gas turbine engine controller design using genetic algorithms. *IEEE Transactions on Industrial Electronics 43*, 5, 583–587.

[26] Choi, Wook-Jin, and Choi, Tae-Sun. Genetic programming-based feature transform and classification for the automatic detection of pulmonary nodules on computed tomography images. *Information Sciences 212* (Dec. 2012), 57–78.

[27] Cleveland, William S., and Devlin, Susan J. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American Statistical Association 83*, 403 (1988), 596–610.

[28] Cornforth, Theodore W., and Lipson, Hod. Inference of hidden variables in systems of differential equations with genetic programming. *Genetic Programming and Evolvable Machines 14*, 2 (June 2013), 155–190.

[29] Cortez, Paulo, Cerdeira, António, Almeida, Fernando, Matos, Telmo, and Reis, José. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems 47*, 4 (Nov. 2009), 547–553.

[30] Cramer, Nichael Lynn. A representation for the adaptive generation of simple sequential programs. In *Proceedings of the First International Conference on Genetic Algorithms* (1985), pp. 183–187.

[31] De Melo, Vincius Veloso. Kaizen programming. ACM Press, pp. 895–902.

[32] Deb, Kalyanmoy, Agrawal, Samir, Pratap, Amrit, and Meyarivan, T. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II. In *Parallel Problem Solving from Nature PPSN VI*, Marc Schoenauer, Kalyanmoy Deb, Gnther Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, Eds., vol. 1917. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000, pp. 849–858.

[33] Dias, Brian G., and Ressler, Kerry J. PACAP and the PAC1 Receptor in Post-Traumatic Stress Disorder. *Neuropsychopharmacology 38*, 1 (Jan. 2013), 245–246.

[34] Dias, Brian G., and Ressler, Kerry J. Parental olfactory experience influences behavior and neural structure in subsequent generations. *Nature Neuroscience 17*, 1 (Jan. 2014), 89–96.

[35] dos Santos, J. A., Ferreira, C. D., Torres, R. da S., Gonçalves, M. A., and Lamparelli, R. A. C. A relevance feedback method based on genetic programming for classification of remote sensing images. *Information Sciences 181*, 13 (July 2011), 2671–2684.

[36] Dunstan, W. J., and Bitmead, R. R. Nonlinear model validation using multiple experiments. In *Proc. of the 42nd IEEE Conf. on Decision and Control* (Maui, Hawaii, USA, 2003).

[37] Elyasaf, Achiya, Hauptman, Ami, and Sipper, Moshe. GA-FreeCell: Evolving solvers for the game of FreeCell. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation* (2011), ACM, pp. 1931–1938.

[38] Eskander, Mona N. Neural network controller for a permanent magnet generator applied in a wind energy conversion system. *Renewable Energy 26*, 3 (July 2002), 463–477.

[39] Espejo, Pedro G., Ventura, Sebastián, and Herrera, Francisco. A survey on the application of genetic programming to classification. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 40*, 2 (2010), 121–144.

[40] Facchinetti, Matteo Luca, de Langre, Emmanuel, and Biolley, Francis. Coupling of structure and wake oscillators in vortex-induced vibrations. *Journal of Fluids and Structures 19*, 2 (2004), 123–140.

[41] Facchinetti, M.L., de Langre, E., and Biolley, F. Coupling of structure and wake oscillators in vortex-induced vibrations. *Journal of Fluids and Structures 19*, 2 (Feb. 2004), 123–140.

[42] Fang, Yongsheng, and Li, Jun. A review of tournament selection in genetic programming. In *Advances in Computation and Intelligence*. Springer, 2010, pp. 181–192.

[43] Ferreira, Candida. Gene Expression Programming: a New Adaptive Algorithm for Solving Problems. *arXiv:cs/0102027* (Feb. 2001). Complex Systems, 13(2): 87-129, 2001.

[44] Fleming, P. J., and Purshouse, R. C. Evolutionary algorithms in control systems engineering: a survey. *Control Engineering Practice 10* (2002), 1223–1241.

[45] Fleming, Paul, Van Wingerden, Jan-Willem, and Wright, Alan D. *Comparing State-space Multivariable Controls to Multi-SISO Controls for Load Reduction of Drivetrain-coupled Modes on Wind Turbines Through Field-testing: Preprint*. National Renewable Energy Laboratory,[National Wind Technology Center, 2011.

[46] Fontana, Alessandro. Epigenetic Tracking: Biological Implications. In *Advances in Artificial Life. Darwin Meets von Neumann*, George Kampis, Istvn Karsai, and Ers Szathmry, Eds., no. 5777 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2011, pp. 10–17.

[47] Franklin, Gene F., Powell, J. David, Krstic, Abbas Emami-NaeiniMiroslav, Kanellakopoulos, Ioannis, and Kokotovic, Petar. *Feedback Control of Dynamic Systems*, seventh ed. Pearson Education, Inc., Upper Saddle River, New Jersey, USA, 2015.

[48] Gathercole, Chris, and Ross, Peter. Dynamic training subset selection for supervised learning in genetic programming. In *Parallel Problem Solving from NaturePPSN III*. Springer, 1994, pp. 312–321.

[49] Gevers, M. A decade of progress in iterative process control design - from theory to practice. *Journal of Process Control 12* (2002), 519–531.

[50] Giraud-Carrier, Christophe. Unifying Learning with Evolution Through Baldwinian Evolution and Lamarckism. In *Advances in Computational Intelligence and Learning*. Springer, 2002, pp. 159–168.

[51] Gonçalves, Ivo, and Silva, Sara. *Balancing learning and overfitting in genetic programming with interleaved sampling of training data*. Springer, 2013.

[52] Gray, Gary J., Murray-Smith, David J., Li, Yun, Sharman, Ken C., and Weinbrenner, Thomas. Nonlinear model structure identification using genetic programming. *Control Engineering Practice 6*, 11 (1998), 1341–1352.

[53] Gregori, Gregor, and Lightbody, Gordon. Nonlinear system identification: From multiple-model networks to Gaussian processes. *Engineering Applications of Artificial Intelligence 21*, 7 (Oct. 2008), 1035–1055.

[54] Gruau, Frederic, and Whitley, Darrell. Adding Learning to the Cellular Development of Neural Networks: Evolution and the Baldwin Effect. *Evolutionary Computation 1*, 3 (Sept. 1993), 213–233.

[55] Guyon, Isabelle, and Elisseeff, André. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res. 3* (Mar. 2003), 1157–1182.

[56] Haber, R, and Unbehauen, Heinz. Structure identification of nonlinear dynamic systemsa survey on input/output approaches. *Automatica 26*, 4 (1990), 651–677.

[57] Hall, Mark, Frank, Eibe, Holmes, Geoffrey, Pfahringer, Bernhard, Reutemann, Peter, and Witten, Ian H. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter 11*, 1 (2009), 10–18.

[58] Harper, Robin. Spatial co-evolution: quicker, fitter and less bloated. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference* (2012), ACM, pp. 759–766.

[59] Hearst, Marti A., Dumais, Susan T, Osman, Edgar, Platt, John, and Scholkopf, Bernhard. Support vector machines. *Intelligent Systems and their Applications, IEEE 13*, 4 (1998), 18–28.

[60] Helmuth, T., Spector, L., and Matheson, J. Solving Uncompromising Problems with Lexicase Selection. *IEEE Transactions on Evolutionary Computation PP*, 99 (2014), 1–1.

[61] Helmuth, Thomas, and Spector, Lee. General Program Synthesis Benchmark Suite. ACM Press, pp. 1039–1046.

[62] Hjalmarsson, H. Iterative feedback tuning - an overview. *Int'l J. of Adaptive Control and Signal Processing 16* (2002), 373–395.

[63] Hoai, Nguyen Xuan, McKay, Robert Ian, Essam, D., and Chau, R. Solving the symbolic regression problem with tree-adjunct grammar guided genetic programming: The comparative results. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on* (2002), vol. 2, IEEE, pp. 1326–1331.

[64] Holland, John H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[65] Holliday, Robin. Epigenetics: a historical overview. *Epigenetics 1*, 2 (2006), 76–80.

[66] Hornby, Gregory S. ALPS: The Age-layered Population Structure for Reducing the Problem of Premature Convergence. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation* (New York, NY, USA, 2006), GECCO '06, ACM, pp. 815–822.

[67] Hrycej, Tomas. *Neurocontrol: Towards an Industrial Control Methodology.* John Wiley & Sons, Inc., 605 Third Ave., New York, NY, 10158-0012, 1997.

[68] Hu, Jian Jun, and Goodman, Erik D. The hierarchical fair competition (HFC) model for parallel evolutionary algorithms. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on* (2002), vol. 1, IEEE, pp. 49–54.

[69] Iba, Hitoshi. Bagging, boosting, and bloating in genetic programming. In *Proceedings of the genetic and evolutionary computation conference* (1999), vol. 2, pp. 1053–1060.

[70] Iba, Hitoshi. Inference of differential equation models by genetic programming. *Information Sciences 178*, 23 (2008), 4453–4468. Special Section: Genetic and Evolutionary Computing.

[71] Iba, Hitoshi, and Sato, Taisuke. Genetic Programming with Local Hill-Climbing. Tech. Rep. ETL-TR-94-4, Electrotechnical Laboratory, 1-1-4 Umezono, Tsukuba-city, Ibaraki, 305, Japan, 1994.

[72] Icke, Ilknur, and Bongard, Josh C. Improving genetic programming based symbolic regression using deterministic machine learning. In *Evolutionary Computation (CEC), 2013 IEEE Congress on* (2013), IEEE, pp. 1763–1770.

[73] Ingalalli, Vijay, Silva, Sara, Castelli, Mauro, and Vanneschi, Leonardo. A Multi-dimensional Genetic Programming Approach for Multi-class Classification Problems. In *Genetic Programming*. Springer, 2014, pp. 48–60.

[74] Iribas, M., and Landau, I.-D. Identification of wind turbines in closed-loop operation in the presence of three-dimensional turbulence wind speed: torque demand to measured generator speed loop. *Wind Energy 12*, 7 (Oct. 2009), 660–675.

[75] Iribas-Latour, M., and Landau, I-D. Identification in closed-loop operation of models for collective pitch robust controller design. *Wind Energy 16*, 3 (Apr. 2013), 383–399.

[76] Ishibuchi, Hisao, Tsukamoto, Noritaka, and Nojima, Yusuke. Evolutionary many-objective optimization: A short review. In *IEEE congress on evolutionary computation* (2008), Citeseer, pp. 2419–2426.

[77] Jablonka, Eva, and Lamb, Marion J. The changing concept of epigenetics. *Annals of the New York Academy of Sciences 981*, 1 (2002), 82–96.

[78] Jackson, J. E. *A User's Guide to Principle Components*. John Wiley & Sons, Inc., 1991.

[79] Jain, A. K., Murty, M. N., and Flynn, P. J. Data Clustering: A Review. *ACM Comput. Surv. 31*, 3 (Sept. 1999), 264–323.

[80] Jeong, Il-kwon, and Lee, Ju-jang. Adaptive simulated annealing genetic algorithm for system identification. *Engineering Applications of Artificial Intelligence 9*, 5 (Oct. 1996), 523–532.

[81] Jonkman, Jason M, and Buhl Jr, Marshall L. FAST users guide. *Golden, CO: National Renewable Energy Laboratory* (2005).

[82] Kaati, G, Bygren, L O, and Edvinsson, S. Cardiovascular and diabetes mortality determined by nutrition during parents' and grandparents' slow growth period. *European Journal of Human Genetics 10*, 11 (Nov. 2002), 682.

[83] Keijzer, Maarten. Improving Symbolic Regression with Interval Arithmetic and Linear Scaling. In *Genetic Programming*, Conor Ryan, Terence Soule, Maarten Keijzer, Edward Tsang, Riccardo Poli, and Ernesto Costa, Eds., no. 2610 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2003, pp. 70–82.

[84] Keijzer, Maarten. Push-forth: a light-weight, strongly-typed, stack-based genetic programming language. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation* (2013), ACM, pp. 1635–1640.

[85] Keijzer, Maarten, and Babovic, Vladan. Genetic programming, ensemble methods and the bias/variance tradeoffIntroductory investigations. In *Genetic Programming*. Springer, 2000, pp. 76–90.

[86] Khalil, Hassan K. *Nonlinear Control*. Pearson Education Inc., 1 Lake St., Upper Saddle River, NJ 07458, 2015.

[87] Killingsworth, N. J., and Krstic, M. Pid tuning using extermum seeking. *IEEE Control Systems Magazine* (Feb. 2006), 70–79.

[88] Kishore, J. K., Patnaik, Lalit M., Mani, V., and Agrawal, V. K. Application of genetic programming for multicategory pattern classification. *Evolutionary Computation, IEEE Transactions on 4*, 3 (2000), 242–258.

[89] Klein, Jon, and Spector, Lee. Genetic programming with historically assessed hardness. *Genetic Programming Theory and Practice VI* (2008), 61–75.

[90] Kommenda, Michael, Kronberger, Gabriel, Affenzeller, Michael, Winkler, Stephan M., and Burlacu, Bogdan. Evolving Simple Symbolic Regression Models by Multi-objective Genetic Programming. In *Genetic Programming Theory and Practice*, vol. XIV of *Genetic and Evolutionary Computation*. Springer, Ann Arbor, MI, 2015.

[91] Kommenda, Michael, Kronberger, Gabriel, Winkler, Stephan, Affenzeller, Michael, and Wagner, Stefan. Effects of constant optimization by nonlinear least squares minimization in symbolic regression. In *GECCO '13 Companion: Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion* (Amsterdam, The Netherlands, 2013), Christian Blum, Enrique Alba, Thomas Bartz-Beielstein, Daniele Loiacono, Francisco Luna, Joern Mehnen, Gabriela Ochoa, Mike Preuss, Emilia Tantar, and Leonardo Vanneschi, Eds., ACM, pp. 1121–1128.

[92] Koza, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA USA, 1992.

[93] Koza, John R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[94] Koza, John R., Bennett III, Forrest H., Andre, David, Keane, Martin A., and Dunlap, Frank. Automated synthesis of analog electrical circuits by means of genetic programming. *Evolutionary Computation, IEEE Transactions on 1*, 2 (1997), 109–128.

[95] Koza, John R., III, Forrest H. Bennett, Andre, David, and Keane, Martin A. Automatic design of analog electrical circuits using genetic programming. In *Intelligent Data Analysis in Science*, Hugh Cartwright, Ed. Oxford University Press, Oxford, 2000, pp. 172–200.

[96] Krawiec, Krzysztof, and Lichocki, Pawel. Using Co-solvability to Model and Exploit Synergetic Effects in Evolution. In *Parallel Problem Solving from Nature, PPSN XI*, Robert Schaefer, Carlos Cotta, Joanna Koodziej, and Gnter Rudolph, Eds. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 492–501.

[97] Krawiec, Krzysztof, and Liskowski, Pawel. Automatic derivation of search objectives for test-based genetic programming. In *Genetic Programming*. Springer, 2015, pp. 53–65.

[98] Krawiec, Krzysztof, and Nawrocki, Mateusz. *Implicit fitness sharing for evolutionary synthesis of license plate detectors*. Springer, 2013.

[99] Krawiec, Krzysztof, and O'Reilly, Una-May. Behavioral programming: a broader and more detailed take on semantic GP. In *Proceedings of the 2014 conference on Genetic and evolutionary computation* (2014), ACM Press, pp. 935–942.

[100] Krstic, Miroslav, Kanellakopoulos, Ioannis, and Kokotovic, Petar. *Nonlinear and Adaptive Control Design*. John Wiley and Sons, New York, NY, USA, 1995.

[101] Kusiak, Andrew, Li, Wenyan, and Song, Zhe. Dynamic control of wind turbines. *Renewable Energy 35*, 2 (Feb. 2010), 456–463.

[102] Kusiak, Andrew, Zheng, Haiyang, and Song, Zhe. Power optimization of wind turbines with data mining and evolutionary computation. *Renewable Energy 35*, 3 (Mar. 2010), 695–702.

[103] La Cava, William, and Danai, Kourosh. Model Structure Adaptation: A Gradient-based Approach. In *ASME 2015 Dynamic Systems and Control Conference* (Columbus, Ohio, Oct. 2015), ASME.

[104] La Cava, William, Danai, Kourosh, and Spector, Lee. Inference of compact nonlinear dynamic models by epigenetic local search. *Engineering Applications of Artificial Intelligence*, 1 (2016). In Press.

[105] La Cava, William, Danai, Kourosh, Spector, Lee, Fleming, Paul, Wright, Alan, and Lackner, Matthew. Automatic identification of wind turbine models using evolutionary multiobjective optimization. *Renewable Energy* (Nov. 2015).

[106] La Cava, William, Danai, Kourosh, Spector, Lee, Fleming, Paul, Wright, Alan D., and Lackner, Matthew. Automated Identification of Closed-Loop Wind Turbine Dynamics via Genetic Programming. In *ASME 2015 Dynamic Systems and Control Conference* (Columbus, Ohio, Oct. 2015), ASME.

[107] La Cava, William, Helmuth, Thomas, Spector, Lee, and Danai, Kourosh. Genetic Programming with Epigenetic Local Search. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (2015), ACM Press, pp. 1055–1062.

[108] La Cava, William, Silva, Sara, Vanneschi, Leonardo, Spector, Lee, and Danai, Kourosh. Multidimensional genetic programming for multiclass classification. *Information Sciences* (2015). Under Review.

[109] La Cava, William, Spector, Lee, and Danai, Kourosh. Epsilon-lexicase selection for regression. In *GECCO 2016* (July 2016), ACM Press. To Appear.

[110] La Cava, William, Spector, Lee, Danai, Kourosh, and Lackner, Matthew. Evolving differential equations with developmental linear genetic programming and epigenetic hill climbing. In *Companion proceedings of the 2014 conference on Genetic and evolutionary computation (GECCO)* (2014), ACM Press, pp. 141–142.

[111] La Cava, William G., and Danai, Kourosh. Gradient-based adaptation of continuous dynamic model structures. *International Journal of Systems Science 47*, 1 (Jan. 2016), 249–263.

[112] Langdon, W. B., and Nordin, J. P. Seeding GP populations. In *Genetic Programming, Proceedings of EuroGP'2000* (Edinburgh, 15-16 Apr. 2000), Riccardo Poli, Wolfgang Banzhaf, William B. Langdon, Julian F. Miller, Peter Nordin, and Terence C. Fogarty, Eds., vol. 1802 of *LNCS*, Springer-Verlag, pp. 304–315.

[113] Langdon, William B. Evolving Data Structures with Genetic Programming. In *ICGA* (1995), pp. 295–302.

[114] Langdon, William B. *Genetic programming and data structures: genetic programming+ data structures= automatic programming!*, vol. 1. Springer Science & Business Media, 2012.

[115] Lequin, O., Gevers M. Mossberg M. Bosmans E., and Triest, L. Iterative feedback tuning of pid parameters: Comparison with classical tuning rules. *Control Engineering Practice 11*, 9 (2003), 1023–1033.

[116] Li, Tao, Zhang, Chengliang, and Ogihara, Mitsunori. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics 20*, 15 (2004), 2429–2437.

[117] Lichman, M. *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences, 2013.

[118] Liskowski, Pawel, Krawiec, Krzysztof, Helmuth, Thomas, and Spector, Lee. Comparison of Semantic-aware Selection Methods in Genetic Programming. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation* (New York, NY, USA, 2015), GECCO Companion '15, ACM, pp. 1301–1307.

[119] Liu, Huan, and Yu, Lei. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering 17*, 4 (Apr. 2005), 491–502.

[120] Liu, Li, Shao, Ling, and Li, Xuelong. Evolutionary compact embedding for large-scale image classification. *Information Sciences 316* (Sept. 2015), 567–581.

[121] Ljung, L. *System Identification: Theory for the User*, 2nd ed. ed., vol. 1. Prentice Hall, 1999.

[122] Ljung, L., and Glad, T. On global identifiability for arbitrary model parametrizations. *Automatica 30*, 2 (1994), 265–275.

[123] Ljung, Lennart. System Identification Toolbox for Use with ${$matlab$}$.

[124] Lohn, Jason D., Hornby, Gregory S., and Linden, Derek S. An evolved antenna for deployment on nasas space technology 5 mission. In *Genetic Programming Theory and Practice II*. Springer, 2005, pp. 301–315.

[125] Loveard, Thomas, and Ciesielski, Victor. Representing classification problems in genetic programming. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on* (2001), vol. 2, IEEE, pp. 1070–1077.

[126] Luke, Sean. *Essentials of Metaheuristics*, 2nd ed. 2013.

[127] Madar, Janos, Abonyi, Janos, and Szeifert, Ferenc. Genetic programming for system identification. In *Intelligent Systems Design and Applications (ISDA 2004) Conference, Budapest, Hungary* (2004).

[128] Mahfoud, Samir W. *Niching methods for genetic algorithms.* PhD thesis, 1995.

[129] Manwell, J. F, McGowan, J. G, and Rogers, Anthony L. *Wind energy explained: theory, design and application.* Wiley, Chichester, U.K., 2009.

[130] Martnez, Yuliana, Naredo, Enrique, Trujillo, Leonardo, and Galvn-Lpez, Edgar. Searching for novel regression functions. In *Evolutionary Computation (CEC), 2013 IEEE Congress on* (2013), IEEE, pp. 16–23.

[131] Massey, Blake, Bowen, Rhys, Griffin, Curtice, and McGarigal, Kevin. A Classification-Tree Analysis of Nesting Habitat in an Island Population of Northern Harriers. *The Condor 110*, 1 (Feb. 2008), 177–183.

[132] McConaghy, Trent. Ffx: Fast, scalable, deterministic symbolic regression technology. In *Genetic Programming Theory and Practice IX*. Springer, 2011, pp. 235–260.

[133] McCusker, J. R., Currier, T., and Danai, K. Improved parameter estimation by noise compensation in the time-scale domain. *Signal Processing 91*, 1 (2011), 72–84.

[134] McKay, R. I. (Bob). An Investigation of Fitness Sharing in Genetic Programming. *The Australian Journal of Intelligent Information Processing Systems 7*, 1/2 (July 2001), 43–51.

[135] Meek, Christopher, Chickering, David Maxwell, and Heckerman, David. Autoregressive Tree Models for Time-Series Analysis. In *SDM* (2002), SIAM, pp. 229–244.

[136] Melin, Patricia, Amezcua, Jonathan, Valdez, Fevrier, and Castillo, Oscar. A new neural network model based on the LVQ algorithm for multi-class classification of arrhythmias. *Information Sciences 279* (Sept. 2014), 483–497.

[137] Messner, William, and Tilbury, Dawn. Control tutorials. http://ctms.engin.umich.edu/CTMS/index.php?aux=Home. Accessed: 2016-1-7.

[138] Miller, Julian F., and Thomson, Peter. Cartesian Genetic Programming. In *Genetic Programming*, Riccardo Poli, Wolfgang Banzhaf, William B. Langdon, Julian Miller, Peter Nordin, and Terence C. Fogarty, Eds., no. 1802 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2000, pp. 121–132.

[139] Moraglio, Alberto, Krawiec, Krzysztof, and Johnson, Colin G. Geometric semantic genetic programming. In *Parallel Problem Solving from Nature-PPSN XII*. Springer, 2012, pp. 21–31.

[140] Muñoz, Luis, Silva, Sara, and Trujillo, Leonardo. M3GP - Multiclass Classification with GP. In *Genetic Programming*. Springer, 2015, pp. 78–91.

[141] Mukherjee, Sayan, Osuna, Edgar, and Girosi, Federico. Nonlinear prediction of chaotic time series using support vector machines. In *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop* (1997), IEEE, pp. 511–520.

[142] Murphy, Kevin P. Dynamic bayesian networks. *Probabilistic Graphical Models, M. Jordan 7* (2002).

[143] Murphy, Kevin P. *Machine learning: a probabilistic perspective*. Adaptive computation and machine learning series. MIT Press, Cambridge, MA, 2012.

[144] Murphy, Kevin Patrick. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002.

[145] Narendra, K. S., and Annaswamy, A. M. *Stable Adaptive Systems*. Prentice Hall, 1989.

[146] Narendra, K. S., and Parthasarathy, K. Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks 1*, 1 (1990), 4–27.

[147] Narendra, Kumpati S., and Parthasarathy, Kannan. Identification and control of dynamical systems using neural networks. *Neural Networks, IEEE Transactions on 1*, 1 (1990), 4–27.

[148] Newton, Sir Isaac. *Sir Isaac Newton's Mathematical Principles of Natural Philosophy and His System of the World*. University of California Press, 1934.

[149] Nguyen, Thanh, Khosravi, Abbas, Creighton, Douglas, and Nahavandi, Saeid. Hidden Markov models for cancer classification using gene expression profiles. *Information Sciences 316* (Sept. 2015), 293–307.

[150] Ni, Xianfeng, Verhaegen, Michel, Krijgsman, Ardjan J., and Verbruggen, Henk B. A new method for identification and control of nonlinear dynamic systems. *Engineering Applications of Artificial Intelligence 9*, 3 (June 1996), 231–243.

[151] Nordin, Peter, Francone, Frank, and Banzhaf, Wolfgang. Explicitly Defined Introns and Destructive Crossover in Genetic Programming. In *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications* (Tahoe City, California, USA, 1995), Justinian P. Rosca, Ed., pp. 6–22.

[152] O'Neill, Michael, Vanneschi, Leonardo, Gustafson, Steven, and Banzhaf, Wolfgang. Open issues in genetic programming. *Genetic Programming and Evolvable Machines 11*, 3-4 (Sept. 2010), 339–363.

[153] Païdoussis, Michael P. *Fluid-Structure Interactions, Vol. 1: Slender Structures and Axial Flow*. San Diego, CA: Academic Press Inc, 1998.

[154] Païdoussis, Michael P, Price, Stuart J, and de Langre, Emmanuel. *Fluid-Structure Interactions*, vol. 1. Cambridge University Press, New York, 2004.

[155] Païdoussis, MP. *Fluid-Structure Interactions, Vol. 2*. Academic Press, London, 2004.

[156] Perkis, Timothy. Stack-based genetic programming. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on* (1994), IEEE, pp. 148–153.

[157] Pham-Gia, T., and Hung, T. L. The mean and median absolute deviations. *Mathematical and Computer Modelling 34*, 78 (Oct. 2001), 921–936.

[158] Platel, M.D., Clergue, M., and Collard, P. Homology gives size control in genetic programming. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03* (Dec. 2003), vol. 1, pp. 281–288 Vol.1.

[159] Poli, Riccardo, McPhee, Nicholas F, and Koza, John R. *A field guide to genetic programming.* [Lulu Press], lulu.com, [S.I.], 2008.

[160] Poli, Riccardo, and McPhee, Nicholas Freitag. Parsimony pressure made easy: Solving the problem of bloat in GP. In *Theory and Principled Methods for the Design of Metaheuristics*, Yossi Borenstein and Alberto Moraglio, Eds., Natural Computing Series. Springer, 2013, pp. 181–204.

[161] Popper, K. *The Logic of Scientific Discovery, Volume 1, Fourth Ed.* Hutchinson & Co. LTD, Great Portland Street, London W1., 1959.

[162] Popper, K. *The Myth of the Framework, Volume 1.* Routledge, 29 West 35th Street, New York, NY 10001., 1994.

[163] Preble, Stefan, Lipson, Michal, and Lipson, Hod. Two-dimensional photonic crystals designed by evolutionary algorithms. *Applied Physics Letters 86*, 6 (2005), 061111.

[164] Quinlan, J Ross. *C4. 5: programs for machine learning.* Elsevier, 2014.

[165] Ren, Fuji, and Sohrab, Mohammad Golam. Class-indexing-based term weighting for automatic text classification. *Information Sciences 236* (July 2013), 109–125.

[166] Reynoso-Meza, Gilberto, Blasco, Xavier, Sanchis, Javier, and MiguelMartnez. Controller tuning using evolutionary multi-objective optimisation: Current trends and applications. *Control Engineering Practice 28* (2014), 58–73.

[167] Rodrguez-Vzquez, Katya, and Fleming, Peter J. Evolution of mathematical models of chaotic systems based on multiobjective genetic programming. *Knowledge and Information Systems 8*, 2 (Oct. 2004), 235–256.

[168] Rodriguez-Vazquez, K., Fonseca, C. M., and Fleming, P. J. 'Identifying the structure of nonlinear dynamic systems using multiobjective genetic programming. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 34*, 4 (July 2004), 531–545.

[169] Ross, Brian J. A Lamarckian evolution strategy for genetic algorithms. *Practical handbook of genetic algorithms: complex coding systems 3* (1999), 1–16.

[170] Ryan, Conor. *Reducing Premature Convergence in Evolutionary Algorithms.* PhD thesis, 1996.

[171] Sadollah, Ali, Eskandar, Hadi, Yoo, Do Guen, and Kim, Joong Hoon. Approximate solving of nonlinear ordinary differential equations using least square weight function and metaheuristic algorithms. *Engineering Applications of Artificial Intelligence 40* (Apr. 2015), 117–132.

[172] Sagha, Hesam, Digumarti, Sundara Tejaswi, Milln, Jos del R., Chavarriaga, Ricardo, Calatroni, Alberto, Roggen, Daniel, and Trster, Gerhard. Benchmarking classification techniques using the Opportunity human activity dataset. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on* (2011), IEEE, pp. 36–40.

[173] Salman, Walid P, Tisserand, Olivier, Toulout, Bruno, and Stewart, MJ. *Forth.* Springer-Verlag New York, Inc., 1985.

[174] Sarpkaya, T. A critical review of the intrinsic nature of vortex-induced vibrations. *Journal of Fluids and Structures 19*, 4 (2004), 389 – 447.

[175] Schmidt, M.D., and Lipson, H. Coevolution of Fitness Predictors. *IEEE Transactions on Evolutionary Computation 12*, 6 (Dec. 2008), 736–749.

[176] Schmidt, Michael, and Lipson, Hod. Comparison of Tree and Graph Encodings As Function of Problem Complexity. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation* (New York, NY, USA, 2007), GECCO '07, ACM, pp. 1674–1679.

[177] Schmidt, Michael, and Lipson, Hod. Distilling free-form natural laws from experimental data. *Science 324*, 5923 (2009), 81–85.

[178] Schmidt, Michael, and Lipson, Hod. Age-fitness pareto optimization. In *Genetic Programming Theory and Practice VIII*. Springer, 2011, pp. 129–146.

[179] Schmidt, Michael D., and Lipson, Hod. Incorporating expert knowledge in evolutionary search: a study of seeding methods. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation* (2009), ACM, pp. 1091–1098.

[180] Schmidt, Michael Douglas. *Machine Science: Automated Modeling of Deterministic and Stochastic Dynamical Systems*. PhD thesis, Cornell University, Ithaca, NY, USA, 2011. AAI3484909.

[181] Schmidt, Michael Douglas, and Lipson, Hod. Automated modeling of stochastic reactions with large measurement time-gaps. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation* (2011), ACM, pp. 307–314.

[182] Seaton, Tom, Brown, Gavin, and Miller, Julian F. Analytic Solutions to Differential Equations under Graph-Based Genetic Programming. In *Genetic Programming*, Anna Isabel Esparcia-Alczar, Anik Ekrt, Sara Silva, Stephen Dignum, and A. ima Uyar, Eds., no. 6021 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2010, pp. 232–243.

[183] Sepulchre, R., Jankovic, M., and Kokotovic, P.V. *Constructive Nonlinear Control*. Springer London, London, UK, 1997.

[184] Seyed-Aghazadeh, B., Budz, C., and Modarres-Sadeghi, Y. The influence of higher harmonic flow forces on the response of a curved circular cylinder undergoing vortex-induced vibration. *Journal of Sound and Vibration*, In Review (2015).

[185] Seyed-Aghazadeh, B., Carlson, D., and Modarres-Sadeghi, Y. The influence of taper ratio on vortex-induced vibrations of tapered cylinders in the crossflow direction. *Journal of Fluids and Structures 53* (2015), 84–95.

[186] Shone, Ronald. *Economic Dynamics: Phase diagrams and their economic application.* Cambridge University Press, 2002.

[187] Silva, Sara, and Costa, Ernesto. Dynamic Limits for Bloat Control: Variations on Size and Depth. In *Genetic and Evolutionary Computation GECCO-2004, Part II* (Seattle, WA, USA, 2004), Kalyanmoy Deb, Riccardo Poli, Wolfgang Banzhaf, Hans-Georg Beyer, Edmund Burke, Paul Darwen, Dipankar Dasgupta, Dario Floreano, James Foster, Mark Harman, Owen Holland, Pier Luca Lanzi, Lee Spector, Andrea Tettamanzi, Dirk Thierens, and Andy Tyrrell, Eds., vol. 3103 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 666–677.

[188] Silva, Sara, Muñoz, Luis, Trujillo, Leonardo, Ingalalli, Vijay, Castelli, Mauro, and Vanneschi, Leonardo. Multiclass Classificatin Through Multidimensional Clustering. In *Genetic Programming Theory and Practice XIII*, vol. 13. Springer, Ann Arbor, MI, May 2015.

[189] Silverman, Bernard W. *Density estimation for statistics and data analysis*, vol. 26. CRC press, 1986.

[190] Smits, Guido F., and Kotanchek, Mark. Pareto-front exploitation in symbolic regression. In *Genetic Programming Theory and Practice II*. Springer, 2005, pp. 283–299.

[191] Spector, Lee. Autoconstructive evolution: Push, pushGP, and pushpop. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)* (2001), pp. 137–146.

[192] Spector, Lee. *Automatic Quantum Computer Programming: a genetic programming approach*, vol. 7. Springer, 2004.

[193] Spector, Lee. Assessment of problem modality by differential performance of lexicase selection in genetic programming: a preliminary report. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion* (2012), pp. 401–408.

[194] Spector, Lee, Clark, David M., Lindsay, Ian, Barr, Bradford, and Klein, Jon. Genetic programming for finite algebras. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation* (2008), ACM, pp. 1291–1298.

[195] Spector, Lee, and Helmuth, Thomas. Uniform linear transformation with repair and alternation in genetic programming. *Genetic Programming Theory and Practice XI, page In preparation. Springer* (2013).

[196] Spector, Lee, and Robinson, Alan. Genetic programming and autoconstructive evolution with the push programming language. *Genetic Programming and Evolvable Machines 3*, 1 (2002), 7–40.

[197] Stanislawska, Karolina, Krawiec, Krzysztof, and Kundzewicz, Zbigniew W. Modeling global temperature changes with genetic programming. *Computers & Mathematics with Applications 64*, 12 (Dec. 2012), 3717–3728.

[198] Storrie-Lombardi, M. C., Lahav, O., Sodr, L., and Storrie-Lombardi, L. J. Morphological Classification of galaxies by Artificial Neural Networks. *Monthly Notices of the Royal Astronomical Society 259*, 1 (Nov. 1992), 8P–12P.

[199] Strogatz, Steven H. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. Westview press, 2014.

[200] Tanev, I, and Yuta, K. Epigenetic programming: Genetic programming incorporating epigenetic learning through modification of histones. *Information Sciences 178*, 23 (Dec. 2008), 4469–4481.

[201] Topchy, Alexander, and Punch, William F. Faster genetic programming based on local gradient search of numeric leaf values. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)* (2001), pp. 155–162.

[202] Towns, John, Cockerill, Timothy, Dahan, Maytal, Foster, Ian, Gaither, Kelly, Grimshaw, Andrew, Hazlewood, Victor, Lathrop, Scott, Lifka, Dave, Peterson, Gregory D., Roskies, Ralph, Scott, J. Ray, and Wilkens-Diehr, Nancy. XSEDE: Accelerating Scientific Discovery. *Computing in Science and Engineering 16*, 5 (2014), 62–74.

[203] Tsoulos, I. G., and Lagaris, I. E. Solving differential equations with genetic programming. *Genetic Programming and Evolvable Machines 7*, 1 (Mar. 2006), 33–54.

[204] Turner, Andrew James, and Miller, Julian Francis. Neutral genetic drift: an investigation using Cartesian Genetic Programming. *Genetic Programming and Evolvable Machines* (May 2015), 1–28.

[205] Turner, Bryan M. Histone acetylation and an epigenetic code. *Bioessays 22*, 9 (2000), 836–845.

[206] USGS. U.s. geological survey (USGS) earth resources observation systems (EROS) data center (EDC), 2012.

[207] Van der Veen, Gijs. Identification of wind energy systems.

[208] van der Veen, Gijs, van Wingerden, Jan-Willem, and Verhaegen, Michel. Global Identification of Wind Turbines Using a Hammerstein Identification Method. *IEEE Transactions on Control Systems Technology 21*, 4 (July 2013), 1471–1478.

[209] Van Wingerden, JW, Houtzager, I, Felici, F, and Verhaegen, M. Closed-loop identification of the time-varying dynamics of variable-speed wind turbines. *International Journal of Robust and Nonlinear Control 19*, 1 (2009), 4–21.

[210] Vanneschi, Leonardo, Archetti, Francesco, Castelli, Mauro, and Giordani, Ilaria. Classification of Oncologic Data with Genetic Programming. *Journal of Artificial Evolution and Applications 2009* (2009), 1–13.

[211] Vladislavleva, E.J., Smits, G.F., and den Hertog, D. Order of Nonlinearity as a Complexity Measure for Models Generated by Symbolic Regression via Pareto Genetic Programming. *IEEE Transactions on Evolutionary Computation 13*, 2 (2009), 333–349.

[212] Von Neumann, John, Burks, Arthur W, et al. Theory of self-reproducing automata. *IEEE Transactions on Neural Networks 5*, 1 (1966), 3–14.

[213] Whitley, Darrell, Gordon, V. Scott, and Mathias, Keith. Lamarckian evolution, the Baldwin effect and function optimization. In *Parallel Problem Solving from Nature (PPSN) III*. Springer, 1994, pp. 5–15.

[214] Widrow, Bernard, and Walach, Eugene. *Adaptive Inverse Control*. Prentice Hall PTR, Upper Saddle River, NJ 07458, 1996.

[215] Wigren, Torbjörn. Recursive prediction error identification and scaling of non-linear state space models using a restricted black box parameterization. *Automatica 42*, 1 (Jan. 2006), 159–168.

[216] Wigren, Torbjörn. Input-output data sets for development and benchmarking in nonlinear identification. *Technical Reports from the department of Information Technology 20* (2010), 2010–020.

[217] Wigren, Torbjörn, and Schoukens, Johan. Three free data sets for development and benchmarking in nonlinear system identification. In *Proc. 2013 Eur. Control Conf.(ECC2013)* (2013), pp. 17–19.

[218] Williamson, C.H.K., and Govardhan, R. VORTEX-INDUCED VIBRATIONS. *Annual Review of Fluid Mechanics 36*, 1 (Jan. 2004), 413–455.

[219] Wright, Alan Duane. *Modern control design for flexible wind turbines.* National Renewable Energy Laboratory, 2004.

[220] Zitzler, Eckart, Laumanns, Marco, and Thiele, Lothar. *SPEA2: Improving the strength Pareto evolutionary algorithm.* Eidgenssische Technische Hochschule Zrich (ETH), Institut fr Technische Informatik und Kommunikationsnetze (TIK), 2001.