SYMBOLIC REGRESSION FOR IDENTIFICATION, PREDICTION, AND CONTROL OF DYNAMICAL SYSTEMS

Kumulative Dissertation
zur Erlangung des akademischen Grades
"doctor rerum naturalium"
(Dr. rer. nat.)
in der Wissenschaftsdisziplin "Computational Physics"

eingereicht an der
Mathematisch-Naturwissenschaftliche Fakultät
der Universität Potsdam

von

MARKUS QUADE

Juni 2018

Markus Quade: *Symbolic Regression for Identification, Prediction, and Control of Dynamical Systems*

SUPERVISORS:
PD Dr. rer. nat. Markus Abel
apl. Prof. Dr. rer. nat. Michael Rosenblum

LOCATION:
Potsdam

TIME FRAME:
June 2018

Published online at the
Institutional Repository of the University of Potsdam:
URN urn:nbn:de:kobv:517-opus4-419790
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus4-419790

ABSTRACT

In the present work, we use symbolic regression for automated modeling of dynamical systems. Symbolic regression is a powerful and general method suitable for data-driven identification of mathematical expressions. In particular, the structure and parameters of those expressions are identified simultaneously.

We consider two main variants of symbolic regression: sparse regression-based and genetic programming-based symbolic regression. Both are applied to identification, prediction and control of dynamical systems.

We introduce a new methodology for the data-driven identification of nonlinear dynamics for systems undergoing abrupt changes. Building on a sparse regression algorithm derived earlier, the model after the change is defined as a minimum update with respect to a reference model of the system identified prior to the change. The technique is successfully exemplified on the chaotic Lorenz system and the van der Pol oscillator. Issues such as computational complexity, robustness against noise and requirements with respect to data volume are investigated.

We show how symbolic regression can be used for time series prediction. Again, issues such as robustness against noise and convergence rate are investigated using the harmonic oscillator as a toy problem. In combination with embedding, we demonstrate the prediction of a propagating front in coupled FitzHugh-Nagumo oscillators. Additionally, we show how we can enhance numerical weather predictions to commercially forecast power production of green energy power plants.

We employ symbolic regression for synchronization control in coupled van der Pol oscillators. Different coupling topologies are investigated. We address issues such as plausibility and stability of the control laws found. The toolkit has been made open source and is used in turbulence control applications.

Genetic programming based symbolic regression is very versatile and can be adapted to many optimization problems. The heuristic-based algorithm allows for cost efficient optimization of complex tasks.

We emphasize the ability of symbolic regression to yield white-box models. In contrast to black-box models, such models are accessible and interpretable which allows the usage of established tool chains.

ZUSAMMENFASSUNG

In der vorliegenden Arbeit nutzen wird symbolische Regression zur automatisierten Modellierung dynamischer Systeme. Symbolische Regression ist eine mächtige und vielseitige Methode, welche zur Daten-getriebenen Identifikation von mathematischen Ausdrücken geeignet ist. Insbesondere werden dabei Struktur und Parameter des gesuchten Ausdrucks parallel ermittelt.

Zwei Varianten der symbolischen Regression werden im Rahmen dieser Arbeit in Betracht gezogen: *sparse regression* und symbolischer Regression basierend auf genetischem Programmieren. Beide Verfahren werden für die Identifikation, Vorhersage und Regelung dynamischer Systeme angewandt.

Wir führen eine neue Methodik zur Identifikation von dynamischen Systemen, welche eine spontane Änderung erfahren, ein. Die Änderung eines Modells, welches mit Hilfe von *sparse regression* gefunden wurde, ist definiert als sparsamste Aktualisierung im Hinblick auf das Modell vor der Änderung. Diese Technik ist beispielhaft am chaotischem Lorenz System und dem van der Pol Oszillator demonstriert. Aspekte wie numerische Komplexität, Robustheit gegenüber Rauschen sowie Anforderungen an Anzahl von Datenpunkten werden untersucht.

Wir zeigen wie symbolische Regression zur Zeitreihenvorhersage genutzt werden kann. Wir nutzen dem harmonischen Oszillator als Beispielmodell, um Aspekte wie Robustheit gegenüber Rauschen sowie die Konvergenzrate der Optimierung zu untersuchen. Mit Hilfe von Einbettungsverfahren demonstrieren wir die Vorhersage propagierenden Fronten in gekoppelten FitzHugh-Nagumo Oszillatoren. Außerdem betrachten wir die kommerzielle Stromproduktionsvorhersage von erneuerbaren Energien. Wir zeigen wie man diesbezügliche die numerische Wettervorhersage mittels symbolischer Regression verfeinern und zur Stromproduktionsvorhersage anwenden kann.

Wir setzen symbolische Regression zur Regelung von Synchronisation in gekoppelten van der Pol Oszillatoren ein. Dabei untersuchen wir verschiedene Topologien und Kopplungen. Wir betrachten Aspekte wie Plausibilität und Stabilität der gefundenen Regelungsgesetze. Die Software wurde veröffentlicht und wird u. a. zur Turbulenzregelung eingesetzt.

Symbolische Regression basierend auf genetischem Programmieren ist sehr vielseitig und kann auf viele Optimierungsprobleme übertragen werden. Der auf Heuristik basierenden Algorithmus erlaubt die effiziente Optimierung von komplexen Fragestellungen.

Wir betonen die Fähigkeit von symbolischer Regression, sogenannte *white-box* Modelle zu produzieren. Diese Modelle sind – im Gegensatz zu *black-box* Modellen – zugänglich und interpretierbar. Dies ermöglicht das weitere Nutzen von etablierten Methodiken.

PUBLICATIONS

This dissertation is based on the following publications and its accompanying software releases. Contributions are listed according to the CRediT (Contributor Roles Taxonomy)[1]; authors are abbreviated by their initials.

Articles

[Qua+16] Markus Quade, Markus Abel, Kamran Shafi, Robert K. Niven, and Bernd R. Noack. "Prediction of dynamical systems by symbolic regression." In: *Phys. Rev. E* 94.1 (July 2016), p. 012214. DOI: `10.1103/physreve.94.012214`.

[Gou+17] Julien Gout, Markus Quade, Kamran Shafi, Robert K. Niven, and Markus Abel. "Synchronization control of oscillator networks using symbolic regression." In: *Nonlinear Dyn* 91.2 (Nov. 2017), pp. 1001–1021. DOI: `10.1007/s11071-017-3925-z`.

[Qua+17] Markus Quade, Julien Gout, and Markus Abel. "Glyph: Symbolic Regression Tools." In: *J Open Res Softw* (Sept. 2017). arXiv: `1803.06226 [cs.MS]`. Submitted.

[Qua+18a] Markus Quade, Markus Abel, J. Nathan Kutz, and Steven L. Brunton. "Sparse identification of nonlinear dynamics for rapid model recovery." In: *Chaos* 28.6 (28 June 2018), p. 063116. DOI: `10.1063/1.5027470`.

[El +18] Yosef El Sayed M., Philipp Oswald, Stephan Sattler, Pradeep Kumar, Rolf Radespiel, Christian Behr, Michael Sinapius, Jan Petersen, Peter Wierach, Markus Quade, Markus Abel, Bernd R. Noack, and Richard Semaan. "Open- and closed-loop control investigations of unsteady Coanda actuation on a high-lift configuration." In: *2018 Flow Control Conference*. American Institute of Aeronautics and Astronautics, June 2018, p. 3684. DOI: `10.2514/6.2018-3684`.

Software

[Qua+18b] Markus Quade, Julien Gout, and Markus Abel. *glyph - Symbolic Regression Tools*. Version 0.3.5. Jan. 2018. URL: `https://github.com/Ambrosys/glyph`. DOI: `10.5281/zenodo.1156654`.

[Qua17] Markus Quade. *cartesian: A lightweight implementation of Cartesian genetic programming with symbolic regression in mind.* Version 0.4.2. Feb. 2017. URL: `https://github.com/ohjeah/cartesian`. DOI: `10.5281/zenodo.291788`.

[Qua18] Markus Quade. *sparsereg - Collection of Modern Sparse Regression Algorithms*. Version 0.8.5. Feb. 2018. URL: `https://github.com/ohjeah/sparsereg`. DOI: `10.5281/zenodo.1173754`.

Contributions by authors:

[Qua+16] Conceptualization, M.Q., M.A. and K.S.; Methodology, M.Q., M.A. and K.S.; Software, M.Q.; Validation, M.Q. and M.A.; Formal Analysis M.Q. and M.A.; Investigation, M.Q., Writing – Original Draft M.Q.; Writing – Review & Editing M.Q., M.A., K.S, R.N., and B.N.; Visualization M.Q.; Supervision

M.A. and K.S.; Project Administration M.A. and R.N.; Funding Acquisition M.A and R.N.

[Gou+17] Conceptualization, M.Q. and M.A.; Methodology, M.Q. and M.A.; Software, J.G. and M.Q.; Validation, J.G. and M.Q.; Formal Analysis J.G., M.Q. and M.A.; Investigation, J.G. and M.Q., Resources M.A.; Writing – Original Draft J.G., M.Q. and M.A.; Writing – Review & Editing M.Q., M.A., R.N., and K.S.; Visualization J.G.; Supervision M.Q. and M.A.; Project Administration M.A.; Funding Acquisition M.A.

[Qua+17] Conceptualization, M.Q. and M.A.; Methodology, M.Q. and M.A.; Software, M.Q. and J.G.; Validation, M.Q.; Investigation, M.Q., Resources M.A.; Writing – Original Draft M.Q.; Writing – Review & Editing M.Q. and M.A.; Visualization J.G.; Supervision M.A.; Project Administration M.A.; Funding Acquisition M.A.

[Qua+18a] Conceptualization, M.Q., M.A., N.K and S.B.; Methodology, M.Q., M.A., N.K and S.B; Software, M.Q.; Validation, M.Q., M.A. and S.B.; Formal Analysis M.Q. and M.A.; Investigation, M.Q., Resources M.A and S.B.; Writing – Original Draft M.Q., M.A. and S.B.; Writing – Review & Editing M.Q., M.A., N.K and S.B; Visualization M.Q. and S.B.; Supervision M.A., N.K and S.B; Project Administration M.A. and S.B.; Funding Acquisition M.Q., M.A., N.K and S.B.

[El +18] Software, M.Q.; Writing – Review & Editing M.Q.; Supervision M.Q.; *Only M.Q. contributions listed.*

[Qua+18b] Software, M.Q., J.G.

[Qua17] Software, M.Q.

[Qua18] Software, M.Q.

ACKNOWLEDGMENTS

I am very grateful to Markus Abel for giving me the opportunity to write my thesis, for mentorship throughout the years, and for many scientific and non-scientific discussions.

I also thank Kamran Shafi for hosting me in Canberra, and Annie and James for letting me stay with them.

Many thanks to Steven Brunton for inviting me to Seattle and for discussions about motif discovery. I am thankful to Suzanne and Bill for providing me a second home in Seattle.

I would like to thank my colleagues for the good teamwork and warming work environment. Special thanks to Thomas Isele and Kevin Webster for proof-reading the manuscript.

Finally, I would like to thank my family and friends, especially my parents, Tino, Olli, Johannes, Richard, and Maria for their continuous support and encouragement.

CONTENTS

ACRONYMS

ADF	Automatic Defined Functions
ANN	Artificial Neural Networks
CGP	Cartesian Genetic Programming
CRC	Collaborative Research Center
ECMWF	European Centre for Medium-Range Weather Forecasts
EMO	Evolutionary Multi-Objective Optimization
ERC	Ephemeral Random Constants
FFX	Fast Function Extraction
GAM	Generalized Additive Models
GA	Genetic Algorithm
GLM	Generalized Linear Model
GP	Genetic Programming
MLC	Machine Learning Control
ML	Machine Learning
NEAT	NeuroEvolution of Augmenting Topologies
NFL	No Free Lunch Theorems
NRMSE	Normalized Root Mean Squared Error
NSGAII	Non-Dominated Sorting Algorithm II
PCA	Principal Component Analysis
PDE	partial differential equation
RMSE	Root Mean Squared Error
SINDy	Sparse Identification of Nonlinear Dynamics

1 INTRODUCTION

1.1 MOTIVATION

Data is one of the pillars of the current human epoch – the information age – with its ever increasing demand and "capacity to store, communicate, and compute information" [2]. It holds the fundamental question of understanding natural and technical processes. Usually, the language we use to formulate an answer is mathematics. Furthermore, the question of time dependencies in data is treated by dynamical systems theory.

Dynamical Systems are located at the core of natural sciences as the mathematical description for a huge variety of systems. This comprises physics, biology and chemistry together with mathematics, but also any engineering field and parts of medicine. In the last 20 years, *complexity science* has emerged using the language of dynamical systems for a deeper understanding of complex systems. Such systems are typically related to huge research initiatives like climatology [3], fluid dynamics with weather prediction [4] and large-scale transportation (airplanes [5], cars [6]), the human brain [7], or traffic flow [8, 9, 10]. Interestingly, the occurrence of "data" as a keyword in talks or session topics at SIAM Conference on Applications of Dynamical Systems grows exponentially [11], highlighting the importance of data in modern research. A dynamical systems description of a real system is always a model which highlights specific properties of the real system.

Typically, a good model arises from thorough evaluation of experimental results, theoretical considerations, and nowadays also, from additional numerical verification. However, for complex systems, as mentioned above, such a procedure is sometimes very hard, due to the involved scaling properties as in turbulence [12, 13], or due to complex coupling between many degrees of freedom in the system [14]. Consequently, the idea to use a computer to assist in modeling is not too far away. Modern search engines, like Google, accumulate the so called "big data" into economically accessible features which eventually end up as banner ads. We use the Machine Learning (ML) approach to reduce data to a set of scientifically accessible functions, which eventually helps the human scientist to find and evaluate models.

In Fig. 1.1 we show a non-comprehensive overview of data-driven methods used for identification, prediction and control of dynamical systems. A method is usually never used in isolation, but instead is paired with data preprocessing steps, *i. e.* feature engineering. Additionally, the estimation method as well as potential preprocessing steps usually have open hyper-parameters. The optimization of hyper-parameters requires meta-learning techniques. Chained preprocessing, estimation and meta-learning steps are called a pipeline in ML jargon. In commercial applications, if not prohibited by computational cost, the estimation method itself is a hyper-parameter of the pipeline.

The methods used to find equations based on data are called symbolic regression, because the equations are symbolically represented in terms of a certain data

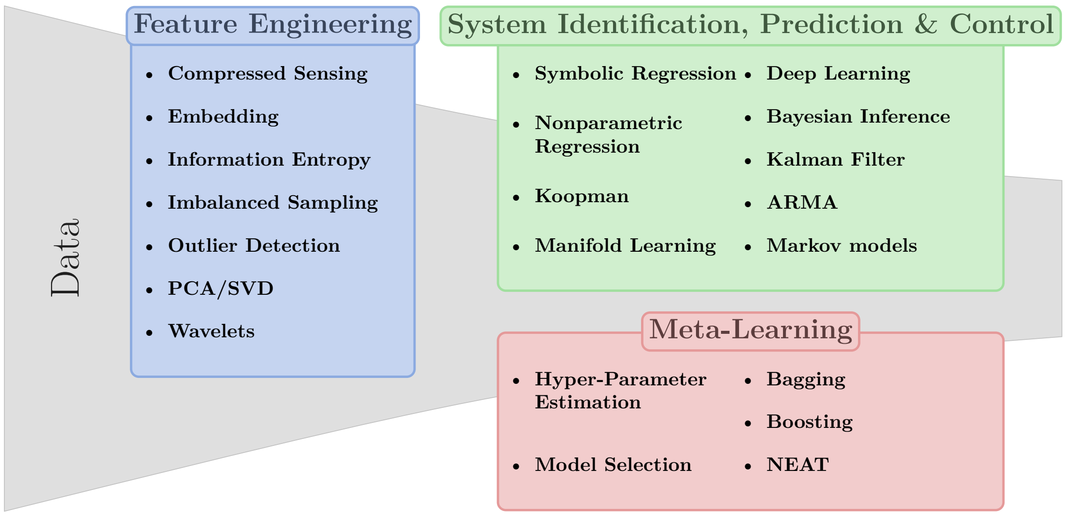


Figure 1.1: Non-comprehensive overview of data-driven modeling methods for identification, prediction and control of dynamical systems.

structure in the computer. Symbolic regression is a mathematical optimization problem: One has to select a particular solution with respect to a cost function that incorporates the various possibilities to fit an equation to data.

In this thesis, we picked symbolic regression as our estimation method. Most ML-methods provide so-called *black-box* models, *i. e.* machines or functions providing an input-output-mapping of data but not allowing model insight. In particular, black-box models often prohibit an analysis with existing domain specific tools and methods.

Contrarily, symbolic regression provides *white-box* models. Provided formulas allow for further symbolic calculus, e. g. computation of gradients for stability analysis or series expansion for first order models. Most importantly, domain experts can continue to rely on their intuition, e. g. when associating terms in an expression with patterns in phase-space. This check for plausibility cannot be provided by black-box models. Symbolic regression is also able to refine expert models based on first principles [15], thus, symbolic regression promotes a high interaction with domain experts.

1.2 STATEMENT OF THE PROBLEM

Since the focus of this thesis is on system identification, prediction and control of dynamical systems, these concepts are explained in the following section.

A dynamical system $\mathcal{S}$ is described by a state vector $\mathbf{y} \in \mathbb{R}^n$. In real system, we have to consider observations reflecting a partial state, *i. e.*

$$\mathbf{x} = \mathbf{g}(\mathbf{y}), \tag{1.1}$$

where $\mathbf{g} : \mathbb{R}^n \mapsto \mathbb{R}^m$ and usually $m < n$.

All methods considered in this thesis are *data-driven*. The collected data $\{\mathbf{x}_i\}_{i=1}^N$ allows the formulation of a regression problem with an objective function $\mathcal{L}(\{\mathbf{x}_i\}_{i=1}^N, \hat{\mathbf{f}})$ that is to be optimized. $\hat{\mathbf{f}}$ is the estimator for the function that is to be discovered.

System Identification

In system identification, we are looking for an estimate $\hat{\mathbf{f}}$ describing the evolution of the state vector $\mathbf{x}$.

$$\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x}) \tag{1.2}$$

The objective function depends on the derivatives, $\mathcal{L}(\dot{\mathbf{x}}, \hat{\mathbf{f}}(\mathbf{x}))$, which have to be inferred from the measurements, as well. See also Section 2.2. For spatial dependencies, we may consider the more general case

$$\partial_t \mathbf{x} = \mathcal{N}(\nabla, \mathbf{x}), \tag{1.3}$$

where $\partial_t \mathbf{x}$ is the partial derivate with respect to time, $\mathcal{N}$ a non-linear operator and ∇ the spatial gradient.

Prediction

In prediction, we are interested in how the state vector evolves in time based on its current and past state and on external forcing:

$$\mathbf{x}(t+\tau) = \mathbf{f}(\mathbf{x}(t), \ldots, t), \tag{1.4}$$

where τ is a time increment. In contrast to system identification, we are not looking for the differential equation describing the evolution of the state, but for its solution, *i.e.* the propagator. To formulate this prediction as a regression problem, we record a time series $\{\mathbf{x}(t)\}_{t=0}^{T}$ and base the objective function on the shifted version $\{\mathbf{x}(t)\}_{t=\tau}^{T+\tau}$ as well as the estimator $\hat{\mathbf{f}}$ applied to the time series $\mathcal{L}(x(t+\tau), \hat{\mathbf{f}}(\mathbf{x}(t), \ldots, t))$.

See also Section 3.4.

(Feedback) Control

In control, we are looking to manipulate the evolution of the system $\mathcal{S}$ towards a desired state. Feedback control, *cf.* Equation 1.5, considers the actuation $\mathbf{a}$ (forcing) based on the current measured state. We introduce a sensing function $\mathbf{s}$ since in practice typically only partial measurements are feasible and full state measurements are prohibitively costly.

$$\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x}) + \mathbf{a}(\mathbf{s}(\mathbf{x})), \tag{1.5}$$

The objective function $\mathcal{L}$ usually includes the success of the control and the time needed to converge to the desired state, but also the cost associated with the actuation and sensing, e. g. energy consumption. With symbolic regression, we are seeking an optimal choice for either $\mathbf{a}$, $\mathbf{s}$, or both simultaneously. See also Section 4.2.1.

1.3 METHODS

1.3.1 *Symbolic Regression*

Symbolic regression can be understood as the most general type of regression analysis. In regression analysis, we try to describe mathematical relationships between variables [16]. Symbolic regression achieves this by searching the space of mathematical expression. This space is usually based on a set of rules or heuristics. Structure and parametric dependencies of the mathematical expression are determined simultaneously.

This understanding of symbolic regressions includes two separate families: sparse nonlinear parametric regression, *cf.* Section 1.3.2, and non-parametric Genetic Programming (GP)-based symbolic regression, *cf.* Section 1.3.3.

Fig. 1.2 depicts a classification chart of symbolic regression. The classification emphasizes the trade-off between computational effort and the algorithm's ability to capture the underlying model complexity.

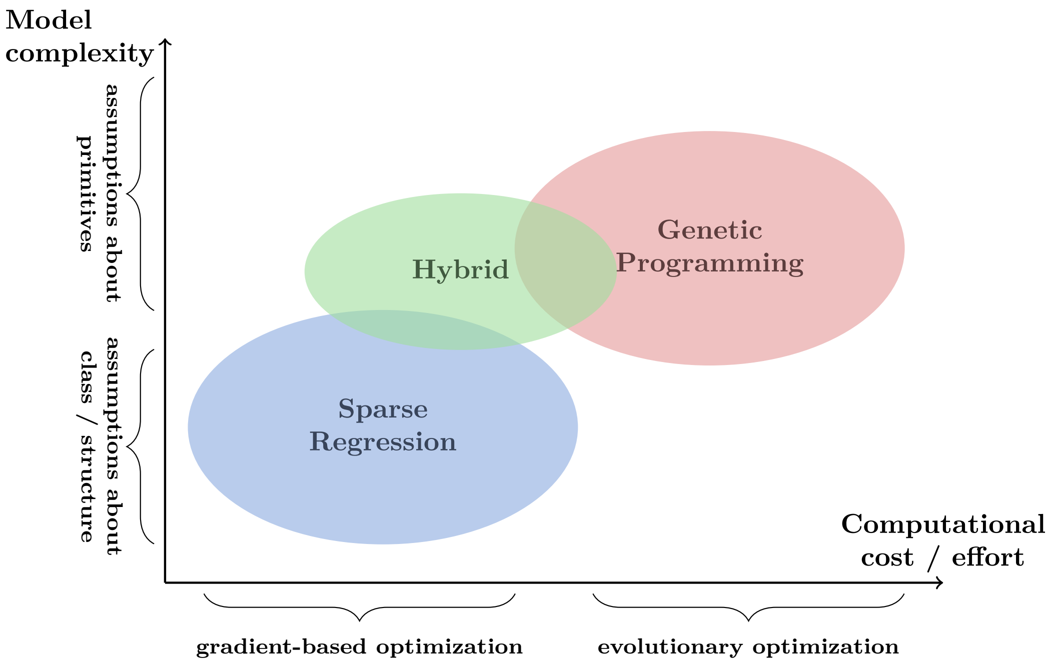

Model complexity
assumptions about primitives
assumptions about class / structure
Hybrid
Genetic Programming
Sparse Regression
Computational cost / effort
gradient-based optimization
evolutionary optimization

Figure 1.2: Overview of existing symbolic regression methods. The diagram shows the model complexity versus the computational effort. The ellipses show the traditional realms of sparse regression and GP-based symbolic regression. Hybrid methods aim to be in the upper left corner.

Sparse regression uses gradient based optimization and therefore has a lower computational cost compared to GP-based symbolic regression. Usually, mmodel complexity is lower as well due to the limitation of more stringent model assumptions.

Ideally, we want our method to be able to produce complex models with low computational effort. This optimum is located in the upper left corner of Fig. 1.2.

Hybrid methods combine ideas from genetic programming and sparse regression, formulating a layered optimization problem. Preferably, hybrid methods are

Pareto optimal in terms of the complexity-effort trade-off. Pareto optimality describes a state of resource allocation towards multiple objectives, such that there is no re-allocation possible which improves an objective without impairing another [17].

1.3.2 *Sparse Regression*

The basic idea of sparse regression is shared by many other machine learning techniques[18]: The data is projected to a high dimensional feature space (feature engineering step) with the goal of finding a low dimensional representation (feature selection step) [19] inside this space. This idea can be realized with generalized linear models (feature engineering) using regularization when optimizing the coefficients such that a few are selected (feature selection). Therefore, *sparsity* refers to the coefficient vector.

With sparse regression, we use the Generalized Linear Model (GLM) *ansatz* [20, 21]

$$\hat{f}(x) = \sum_i \xi_i \phi_i(x), \tag{1.6}$$

where ϕ_i are the set of nonlinear candidate functions, typically drawn from a function family such as polynomials or Fourier series [22], or described by a more complex set of rules [23]. Often we refer to the set of candidate functions $\{\phi_i\}$ as the library. The coefficients ξ_i are determined by minimizing the objective function $\mathcal{L}(\xi_i)$ using gradient based optimization, *i.e.* least squares regression [24, 20, 21]. With a poor choice of the candidate functions ϕ_i, e.g. if library functions are non-orthogonal and/or overdetermined, the GLM *ansatz* is prone to overfitting. However, overfitting can be counteracted by optimizing a regularized version of the objective function

$$\mathcal{L}' = \mathcal{L} + \lambda \mathcal{R}(\xi) \tag{1.7}$$

where $\mathcal{R}(\xi)$ is a penalty function prohibiting the choice of large coefficients ξ_i and λ is a hyper-parameter balancing complexity and sparsity of the solution. Usually, least squares is used $\mathcal{L} = \frac{1}{2n_{\text{samples}}} \|x - \hat{x}\|_2$[1], where n_{samples} is the sample size. Normalizing the least squares error on sample size makes values for λ comparable between different samples, e.g. training and testing data. Choices for the regularization $\mathcal{R}(\xi)$ include (*cf.* [19] for a comprehensive list)

- Lasso regression $\|\xi\|_1$ [24]
- Ridge regression $\|\xi\|_2$ [25]
- Elastic Net regression $\rho \|\xi\|_1 + (1-\rho) \|\xi\|_2$ [26]
- Group Lasso, Sparse Group Lasso[27, 28]

Additionally, thresholding can be used. With thresholding, the regression problem is solved iteratively, disregarding certain library functions $\phi_i(x)$ if their corre-

1 $\|x\|_j = (\sum_a \|x_a\|)^{1/j}$

sponding coefficient does not satisfy specific constraints. Options include sequential thresholding [29] or bootstrapped adaptive thresholding [30]. Thresholding can be combined with regularization.

A big drawback and criticism of sparse regression concerns the effect of selecting the right library functions [31, 32, 33, 20]. If the target function f is embedded in $\{\phi_i\}$ and thus an optimal solution exists, aggressive promotion of sparsity using regularization and thresholding will lead to the desired *sparse* result, *cf.* Fig. 1.3. However, a poor choice of library functions can lead to an under- or overfit solution, which is possibly not sparse at all. A good choice for $\{\phi_i\}$ often requires partial knowledge of the solution itself. This leads to exceptional performance of sparse regression in textbook benchmarks while *sparse* results in real-world problems do not show the same degree of success. More details on sparse regression

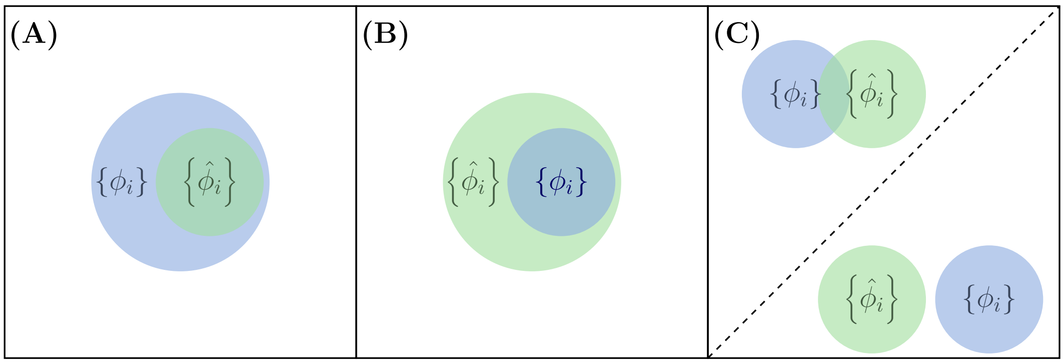


Figure 1.3: Common situations in library estimation in sparse regression. $\{\phi_i\}$ denotes the function space of the ground truth, $\{\hat{\phi}_i\}$ the models estimate. (A) The estimate lies within the ground truth. Even a non sparse fit will underfit. (B) The ground truth lies within the model estimate. The optimal solution will be a sparse fit. This is the desired case. (C) Model estimate and ground truth overlap only partly or are completely distinct. The optimal fit in this case is either sparse and under- or overfit.

can be found in 3.2.1 and Section 2.2.

1.3.3 *Genetic Programming-Based Symbolic Regression*

GP is a bio-inspired optimization heuristic [34]. In contrast to sparse regression, *cf.* Section 1.3.2, it is a fitness guided or *evolutionary* optimization method. Evolution is the interplay of random change and natural selection in a population [35]. In evolutionary optimization, no gradients of the objective function are available. Instead, the objective function is probed by candidate solutions and the search is implicitly guided towards the optimum as *fitter* candidate solutions are preferred over less fit ones. On small scales, *i. e.* considering only a few iterations of probing with and selecting fit candidate solutions, evolutionary optimization works only slightly better than random search [36]. The advantage will only accumulate on larger scales which makes evolutionary optimization inherently computationally expensive and thus slow. Also, even if a solutions exists, the stochastic nature of evolutionary optimization does not guarantee convergence to that solution. On the other hand GP offers great flexibility considering both the structure of the desired solution as well as the objective function.

This makes GP suitable for symbolic regression in system identification [37, 15, 38], prediction [39, 40, 41] and control [42, 43, 44]. A detailed description of the algorithm and its application to symbolic regression can be found in Section 3.2.1 and Section 4.2.2. Further details on representing symbolic formulae in GP are listed in Appendix A.

GP has applications outside the scope of this thesis too, including discovery of conservation laws [45], NeuroEvolution of Augmenting Topologies (NEAT) [46] or the direct optimization of machine code [47], e. g. for optimization of a controller itself instead of providing an actuation law.

1.3.4 *Hybrid methods*

Hybrid methods provide a means of automating feature engineering, *i. e.* feature learning [48, 49].

The sequential feature learning approach has two steps: i) find features correlated to the target, e. g. optimizing $\mathcal{L}_y = \text{corr}(\phi(x), y)$, and ii) perform the regression analysis on the enhanced input space. There are two variants; either GP or sparse regression [50] can be used for the feature learning and sparse regression or GP for the regression respectively. The feature learning step provides a weak estimator. In practice, this is realized using shallow trees or low order polynomials for GP or the library functions in sparse regression respectively. Under the assumption that the desired (strongly non-linear) function can be expressed as the concatenation of two weakly nonlinear function, feature learning will provide a runtime speedup.

In library optimization, we try to tackle the biggest criticism of sparse regression by optimizing the library functions $\{\phi_i\}$ in addition to the coefficients ξ_i, *cf.* Equation 1.6. The set of library functions is optimized by GP. The fitness of library functions is usually based on their *importance* [51], *i. e.* a measure based on the corresponding coefficients ξ_i across different hyper-parameters. Noteworthy implementations include Evolutionary Feature Synthesis [51] and Feature Engineering Wrapper [52].

Feature space transformation is conducted as a layered optimization problem. The simplest implementation is symbolic constants [53], see also Section A.2. In *Linear Feature Space Transformations* [54] a special terminal node is introduced for GP-based symbolic regression: instead of input nodes referring to specific input variables x_i only a single node representing a linear transformation of all inputs $\mathbf{x}' = \mathbf{w}^T\mathbf{x} + \beta$ is used. The weights $\mathbf{w}$ and offset β are optimized using sparse regression either globally (shared across terminal nodes for each expression) or locally.

1.3.5 *Related Work*

Feature Engineering

Feature engineering is a critical step in ML tasks [55]. It can be conducted sequentially or be part of the optimization method itself. Feature engineering changes the size of the input data matrix. Measurements or transformations of measurements can be added or discarded. Principal Component Analysis (PCA) [56] is commonly used to create a reduced set of orthogonal features. Embedding techniques and wavelets can be used to enrich the feature set and to cope with time dependencies. In the context of dynamical systems, Takens' theorem [57] gives the conditions under which a dynamical system can be identified from time-delayed observations.

Heuristic-based feature engineering is sometimes part of estimators [23, 51, 52], see also Section 1.3.4. The first layers in deep neural network architectures can also be interpreted as feature engineering layers. With increasing computational capabilities, automated feature engineering becomes feasible [58, 59].

The importance of feature engineering can be informally summarized by the "garbage in, garbage out"-idiom [60, 61].

Koopman Theory

In related works, the Koopman operator perspective on dynamical systems [62, 63, 64, 65, 66] promises data-driven, linear representations of nonlinear systems to enable optimal nonlinear estimation [67] and control [66].

The Koopman operator $\mathcal{K}$ is a possibly infinite dimensional linear operator which evolves measurements $\mathbf{g}(\mathbf{x})$ of the dynamical system in time

$$\frac{d}{dt}\mathbf{g} = \mathcal{K}\mathbf{g}. \tag{1.8}$$

The Koopman operator framework provides a linear embedding of nonlinear dynamics. Koopman models have recently been used to improve model predictive control for strongly nonlinear systems [68], and eigenfunctions of the Koopman operator provide intrinsic coordinates for optimal nonlinear control [69]. Importantly, there are deep connections between Koopman analysis to physics first-principles, with conserved quantities and known constraints either discovered or encoded in the identified models. In [70], conservation laws are discovered from measurements of a physical system, and these conserved quantities correspond to eigenfunctions of the Koopman operator, which may be useful for designing nonlinear controllers that exploit the dynamics instead of fighting them [66]. The Sparse Identification of Nonlinear Dynamics (SINDy) architecture has recently been used to identify these Koopman coordinates purely from data [66, 69].

Deep Learning

Deep learning is a family of optimization methods based on artificial neural networks with many layers as underlying architectures [71]. Deep learning is very versatile; applications include natural language processing, audio recognition, computer vision, and robotics. Different architectures have been popularized in different domains, e. g. convolutional neural networks in image recognition tasks. Recurrent neural networks are of particular interest in dynamical systems. Deep learning has been applied to system identification [71, 72, 73, 74, 75, 76, 77, 78], time series prediction [79, 80, 81] and control [82, 83].

Meta-Learning

Optimization methods usually have one or more hyper-parameters tuning the models' ability to describe data. Hyper-parameters can either be discrete or continuous. Model performance can depend critically on specific values of hyper-parameters.

Model selection is the task of selecting a model from a set of candidate models. Candidate models can be of the same family based on different hyper-parameters. Model selection is often motivated by Occam's razor ("law of parsimony")[84], *i. e.* if two models have the same predictive capabilities, the less complex model is preferred [36].

A weak estimator correlates only slightly with the target, whereas strong estimators correlate arbitrarily well. Bagging [85] and boosting [86] are meta-estimators aggregating several weak estimators, e. g. by a weighted average. We assume that this leads to a more robust strong estimator. The weak estimators can be of the same family, but trained with different hyper-parameters or different data, or of a different family. In symbolic regression, one can create weak estimators forcing a low complexity of the models, e. g. via high regularization or by using shallow trees.

1.4 OUTLINE

The remainder of this thesis is organized as follows:

- In Chapter 2 we use sparse regression for identification of dynamical systems and introduce a novel paradigm of abrupt change detection based on Lyapunov time estimation.
- In Chapter 3 we apply symbolic regression to time series prediction. The examples range from toy problems (harmonic oscillator) to a real world scenario (solar power production forecast). The latter underlines the commercial relevance of the methods presented in this thesis.
- In Chapter 4 we use GP-based symbolic regression for synchronization control in networks of coupled oscillators.
- In Chapter 5 we describe our implementation of GP-based symbolic regression for control geared towards Machine Learning Control (MLC).
- Finally, in Chapter 6 we conclude, summarizing the relevance of the presented work and show possible directions for future research.

Additionally, technical details on GP-based symbolic regression regarding its representation can be found in Appendix A. The software developed and used alongside the scientific journal publications is briefly described in Appendix B. Appendix C gives a glimpse on preliminary experimental results of applying symbolic regression to control a cross-flow turbine.

2 SPARSE IDENTIFICATION OF DYNAMICAL SYSTEMS FOR RAPID MODEL RECOVERY

by Markus Quade[1, 2], Markus Abel[1, 2], J. Nathan Kutz[3], Steven L. Brunton[4]

[1] Universität Potsdam, Institut für Physik und Astronomie, Karl-Liebknecht-Straße 24/25, 14476 Potsdam, Germany
[2] Ambrosys GmbH, David-Gilly Straße 1, 14469 Potsdam, Germany
[3] Department of Applied Mathematics, University of Washington, Seattle, WA 98195, USA
[4] Department of Mechanical Engineering, University of Washington, Seattle, WA 98195, USA

This paper has been published in Chaos. Dynamical systems modeling is a cornerstone of modern mathematical physics and engineering. The dynamics of many complex systems (e. g. , neuroscience, climate, epidemiology, etc.) may not have first-principles derivations, and researchers are increasingly using data-driven methods for system identification and the discovery of dynamics. Related to discovery of dynamical systems models from data is the recovery *of these models following abrupt changes to the system dynamics. In many domains, such as aviation, model recovery is mission critical, and must be achieved rapidly and with limited noisy data. This paper leverages recent advances in sparse optimization to identify the fewest terms required to recover a model, introducing the concept of* parsimony of change. *In other words, many abrupt system changes, even catastrophic bifurcations, may be characterized with relatively few changes to the terms in the underlying model. In this work, we show that sparse optimization enables rapid model recovery that is faster, requires less data, is more accurate, and has higher noise robustness than the alternative approach of re-characterizing a model from scratch.*

Abstract

Big data has become a critically enabling component of emerging mathematical methods aimed at the automated discovery of dynamical systems, where first principles modeling may be intractable. However, in many engineering systems, abrupt changes must be rapidly characterized based on limited, incomplete, and noisy data. Many leading automated learning techniques rely on unrealistically large data sets and it is unclear how to leverage prior knowledge effectively to re-identify a model after an abrupt change. In this work, we propose a conceptual framework to recover parsimonious models of a system in response to abrupt changes in the low-data limit. First, the abrupt change is detected by comparing the estimated Lyapunov time of the data with the model prediction. Next, we apply the Sparse Identification of Nonlinear Dynamics (SINDy) regression to update a previously identified model with the fewest changes, either by addition, deletion, or modification of existing model terms. We demonstrate this sparse model recovery on several examples for abrupt system change detection in periodic and chaotic dynamical systems. Our examples show that sparse updates to a previously identified model perform better with less data, have lower runtime complexity, and are less sensitive to noise than identifying an entirely new model. The proposed abrupt-SINDy architecture provides a new paradigm for the rapid and efficient recovery of a system model after abrupt changes.

Keywords— Dynamical systems, Chaos, Data-driven models, Machine learning, Sparse optimization

2.1 INTRODUCTION

The data-driven discovery of physical laws and dynamical systems is poised to revolutionize how we model, predict, and control physical systems. Advances are driven by the confluence of big data, machine learning, and modern perspectives on dynamics and control. However, many modern techniques in machine learning (e. g. , neural networks) often rely on access to massive data sets, have limited ability to generalize beyond the attractor where data is collected, and do not readily incorporate known physical constraints. These various limitations are framing many state-of-the-art research efforts around learning algorithms [P1], especially as it pertains to generalizability, limited data and *one-shot learning* [P2, P3, P4]. Such limitations also frame the primary challenges and limitations associated with data-driven discovery for real-time control of strongly nonlinear, high-dimensional, multi-scale systems with abrupt changes in the dynamics. Whereas traditional methods often require unrealistic amounts of training data to produce a viable model, this work focuses on methods that take advantage of prior experience and knowledge of the physics to dramatically reduce the data and time required to characterize dynamics. Our methodology is similar in philosophy to the machine learning technique of *transfer learning* [P5], which allows networks trained on one task to be efficiently adapted to another task. Our architecture is designed around the goal of rapidly extracting parsimonious, nonlinear dynamical models that identify only the fewest important interaction terms so as to avoid overfitting.

There are many important open challenges associated with data-driven discovery of dynamical systems for real-time tracking and control. When abrupt changes occur in the system dynamics, an effective controller must rapidly characterize and compensate for the new dynamics, leaving little time for recovery based on limited data [P6]. The primary challenge in real-time model discovery is the reliance on large quantities of training data. A secondary challenge is the ability of models to generalize beyond the training data, which is related to the ability to incorporate new information and quickly modify the model. Machine learning algorithms often suffer from overfitting and a lack of interpretability, although the application of these algorithms to physical systems offers a unique opportunity to enforce known symmetries and physical constraints (e. g. conservation of mass). Inspired by biological systems, which are capable of extremely fast adaptation and learning based on very few trials of new information [P7, P8, P9], we propose model discovery techniques that leverage an *experiential framework*, where known physics, symmetries, and conservation laws are used to rapidly infer model changes with limited data.

2.1.1 *Previous work in system identification*

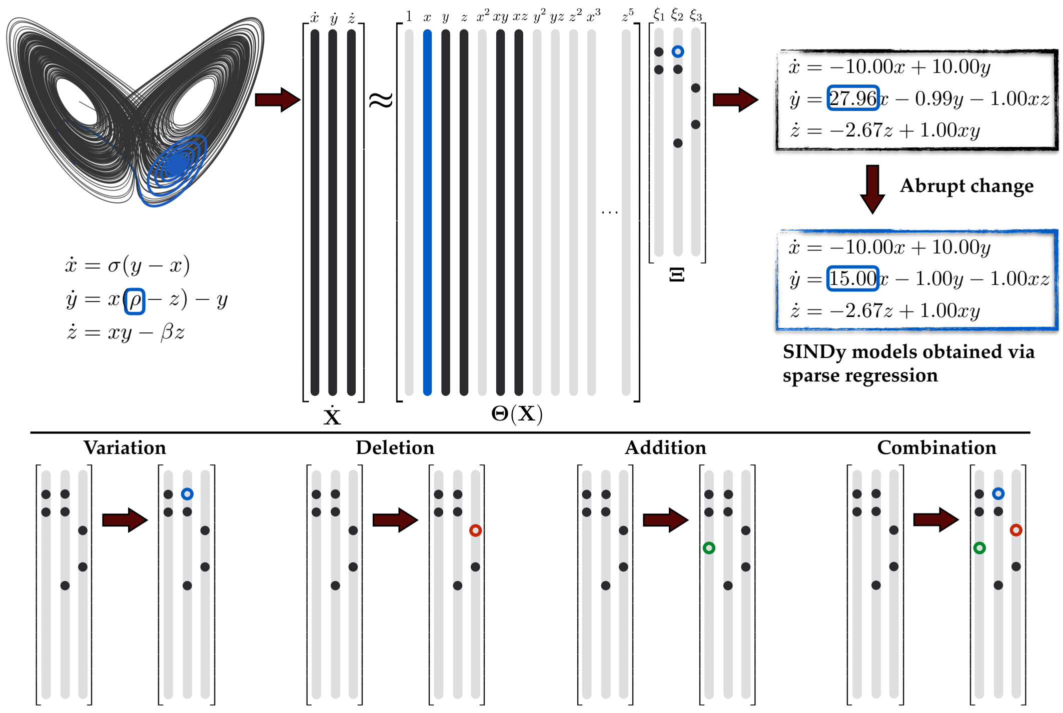
There are a wealth of regression techniques for the characterization of system dynamics from data, with varying degrees of generality, accuracy, data requirements, and computational complexity. Classical linear model identification algorithms include Kalman filters [P10, P11, P12], the eigensystem realization algorithm (ERA) [P13], dynamic mode decomposition (DMD) [P14, P15, P16, P17], and autoregressive moving average (ARMA) models [P18, P19], to name only a few. The resulting linear models are ideal for control design, but are unable to capture the underlying nonlinear dynamics or structural changes. Increasingly, machine learning is being used for nonlinear model discovery. Neural networks have been used for decades to identify nonlinear systems [P20], and are experiencing renewed interest because of the ability to train deeper networks with more data [P1, P21, P22, P23, P24, P25] and the promise of transformations that linearize dynamics via the Koopman operator [P26, P27]. Neural networks show good capacity to recover the dynamics in a so-called "model-free" way [P28, P29]. These methods are also known as "reservoir computers", "liquid state machines", or "echo state networks", depending on the context. However, a real-time application is unrealistic, and the output is generally not analytically interpretable. In another significant vein of research, genetic programming [P30, P31] is a powerful bio-inspired method that has successfully been applied to system identification [P32, P33, P34, P35], time-series prediction [P36, P37] and control [P38, P39]. However, evolutionary methods in their pure form, including genetic programming, are computationally complex and thus are not suitable for real-time tracking.

Recently, *interpretability* and *parsimony* have become important themes in nonlinear system identification [P32, P33]. A common goal now is to identify the fewest terms required to represent the nonlinear structure of a dynamical system model while avoiding overfitting [P40]. Symbolic regression methods [P41, P33, P42, P40] are generally appealing for system identification of structural changes, although they may need to be adapted to the low-data limit and for faster processing time. Nonparametric additive regression models [P43, P44, P45] require a backfitting loop which allows general transformations, but may be prohibitively slow for real-time applications. Generalized linear regression methods are slightly less general but can be brought to a fast evaluation and sparse representation [P42, P40]. These leading approaches to identify dynamical equations from data usually rely on past data and aim at reliable reproduction of a stationary system, i.e. when the underlying equations do not change in the course of time [P33, P45, P40].

2.1.2 *Contributions of this work*

In this work, we develop an adaptive modification of the sparse identification of nonlinear dynamics (SINDy) algorithm [P40] for real-time recovery of a model following abrupt changes to the system dynamics. We refer to this modeling framework as *abrupt-SINDy*. Although this is not the only approach for real-time change detection and recovery, parsimony and sparsity are natural concepts to track abrupt changes, focusing on the fewest modifications to an existing model. SINDy already requires relatively small amounts of data [P46], is based on fast regression techniques, and has been extended to identify partial differential equations (PDEs)[P47, P48], to include known constraints and symmetries [P49], to work with limited measurements [P50] and highly corrupted and noisy data [P51, P52], to include control inputs [P53, P46], and to incorporate information criteria to assess the model quality [P54], which will be useful in abrupt model recovery.

Here, we demonstrate that the abrupt-SINDy architecture is capable of rapidly identifying sparse changes to an existing model to recover the new dynamics following an abrupt change to the system. The first step in the adaptive identification process is to detect a system change using divergence of the prediction from measurements. Next, an existing model is updated with sparse corrections, including parameter variations, deletions, and additions of terms. We show that identifying sparse model changes from an existing model requires less data, less computation, and is more robust to noise than identifying a new model from scratch. Further, we attempt to maintain a critical attitude and caveat limitations of the proposed approach, highlighting when it can break down and suggesting further investigation. The overarching framework is illustrated in Fig. 2.1.



$\dot{x} = \sigma(y - x)$
$\dot{y} = x(\boxed{\rho} - z) - y$
$\dot{z} = xy - \beta z$
$\dot{x}$ $\dot{y}$ $\dot{z}$
$\approx$
1 x y z x^2 xy xz y^2 yz z^2 x^3 … z^5
ξ_1 ξ_2 ξ_3
$\dot{\mathbf{X}}$
$\Theta(\mathbf{X})$
Ξ
$\dot{x} = -10.00x + 10.00y$
$\dot{y} = \boxed{27.96}x - 0.99y - 1.00xz$
$\dot{z} = -2.67z + 1.00xy$
Abrupt change
$\dot{x} = -10.00x + 10.00y$
$\dot{y} = \boxed{15.00}x - 1.00y - 1.00xz$
$\dot{z} = -2.67z + 1.00xy$
SINDy models obtained via sparse regression
Variation
Deletion
Addition
Combination

Figure 2.1: Schematic overview of abrupt-SINDy method. (top) Illustration of single variation in a term, giving rise to an abrupt change in the dynamics, from black to blue. (bottom) Canonical sparse changes, including parameter variation, addition, deletion, or a combination. The data and results for the top panel are from the example in Section 2.4.1 below.

2.2 STATE OF THE ART

Recently, sparse regression in a library of candidate nonlinear functions has been used for sparse identification of nonlinear dynamics (SINDy) to efficiently identify a sparse model structure from data [P40]. The SINDy architecture bypasses an intractable brute-force search through all possible models, leveraging the fact that many dynamical systems of the form

$$\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x}) \tag{2.1}$$

have dynamics $\mathbf{f}$ that are sparse in the state variable $\mathbf{x} \in \mathbb{R}^n$. Such models may be identified using a sparsity-promoting regression [P55, P56, P57] that penalizes the number of nonzero terms ξ_{ij} in a generalized linear model:

$$\hat{f}_i(\mathbf{x}) = \sum_{j=1}^{p} \xi_{ij}\theta_j(\mathbf{x}), \tag{2.2}$$

where $\theta_j(\mathbf{x})$ form a set of nonlinear candidate functions. The candidate functions may be chosen to be polynomials, trigonometric functions, or a more general set of functions [P40, P42]. With poor choice of the candidate functions θ_j, i.e. if library functions are non-orthogonal and/or overdetermined, the SINDy approach may fail to identify the correct model.

Sparse models may be identified from time-series data, which are collected and formed into the data matrix

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots \mathbf{x}_m \end{bmatrix}^T . \tag{2.3}$$

We estimate the time derivatives using a simple forward Euler finite-difference scheme, i.e. the difference of two consecutive data, divided by the time difference:

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{x}}_1 & \dot{\mathbf{x}}_2 & \cdots \dot{\mathbf{x}}_m \end{bmatrix}^T . \tag{2.4}$$

This estimation procedure is numerically ill-conditioned if data are noisy, although there are many methods to handle noise which work very well if used correctly [P58, P59]. Noise-robust derivatives were investigated in the original SINDy algorithm [P40]. Next, we consider a library of candidate nonlinear functions $\mathbf{\Theta}(\mathbf{X})$, of the form

$$\mathbf{\Theta}(\mathbf{X}) = \begin{bmatrix} \mathbf{1} & \mathbf{X} & \mathbf{X}^2 & \cdots & \mathbf{X}^d & \cdots & \sin(\mathbf{X}) & \cdots \end{bmatrix} . \tag{2.5}$$

Here, the matrix $\mathbf{X}^d$ denotes a matrix with column vectors given by all possible time-series of d-th degree polynomials in the state $\mathbf{x}$. The terms in $\mathbf{\Theta}$ can be functional forms motivated by knowledge of the physics. Within the proposed work, they may parameterize a piecewise-affine dynamical model. Following best practices of statistical learning [P56], to preprocess, we mean-subtract and normalize each column of $\mathbf{\Theta}$ to have unit variance. The dynamical system can now be represented in terms of the data matrices as

$$\dot{\mathbf{X}} \approx \mathbf{\Theta}(\mathbf{X})\Xi. \tag{2.6}$$

The coefficients in the column Ξ_k of Ξ determine the active terms in the k-th row of Eq. (2.2). A parsimonious model has the fewest terms in Ξ required to explain the data. One option to obtain a sparse model is via convex ℓ_1-regularized regression:

$$\Xi = \underset{\Xi'}{\operatorname{argmin}} \|\dot{\mathbf{X}} - \mathbf{\Theta}(\mathbf{X})\Xi'\|_2 + \gamma\|\Xi'\|_1. \tag{2.7}$$

The hyper parameter γ balances complexity and sparsity of the solution. Sparse regression, such as LASSO [P55] and sequential thresholded least-squares [P40], improves the robustness of identification for noisy overdetermined data, in contrast to earlier methods [P60] using compressed sensing [P61, P62]. Other regularization schemes may be used to improve performance, such as the elastic net regression [P63].

In this paper we use the sequentially thresholded ridge regression [P47], which iteratively solves the ridge regression

$$\Xi = \underset{\Xi'}{\operatorname{argmin}} \|\dot{\mathbf{X}} - \mathbf{\Theta}(\mathbf{X})\Xi'\|_2 + \alpha\|\Xi'\|_2. \tag{2.8}$$

and then thresholds any coefficient that is smaller than γ. The procedure is repeated on the non-zero entries of Ξ until the model converges. The convergence of the SINDy architecture has been discussed in [P64]. After a sparse model structure has been identified in normalized coordinates, it is necessary to regress onto this sparse structure in the original unnormalized coordinates. Otherwise, non-physical constant terms appear when transforming back from normalized coordinates due to the mean-subtraction.

In La Cava et al. [P35] the authors pursue a complementary although more computationally intensive idea of adaptive modeling in the context of generalized linear models. Starting from an initial guess for the model, a brute force search is conducted to scan a larger set of candidate functions $\theta \to \theta\theta'^{\gamma}$, where θ' are multiplicative extensions to the initial set of candidate functions and γ are real valued exponents. The intended use of this method is the refinement of first-principle based models by discovery of coupling terms. It is possible to combine this refinement with our proposed scheme for dealing with abrupt changes. In addition, sparse sensors [P65] and randomized algorithms [P66] may improve speed.

2.3 METHODS

The viewpoint of sparsity extends beyond model discovery, and we propose to extend SINDy to identify systems undergoing abrupt changes. It may be the case that abrupt model changes will only involve the addition, deletion, or modification of a few terms in the model. This is a statement of the *parsimony of change*, and indicates that we can use sparse regression to efficiently identify the new or missing terms with considerably less data than required to identify a new model from scratch. In general, each additional term that must be identified requires additional training data to distinguish between joint effects. Thus, having only a few changes reduces the amount of data required, making the model recovery more rapid. This section will describe a procedure that extends SINDy to handle three basic types of model changes:

i) **Variation of a term.** If the structure of the model is unchanged and only the parameters vary, we will perform least-squares regression on the known structure to identify the new parameters. This is computationally fast, and it is easy to check if the model explains the new dynamics, or if it is necessary to explore possible additions or deletions of terms.

ii) **Deletion of a term.** If the model changes by the removal of a few terms, then SINDy regression can be applied on the sparse coefficients in order to identify which terms have dropped out.

iii) **Addition of a term.** If a term is added, then SINDy regression will find the sparsest combination of inactive terms that explain the model error. Since least squares regression scales asymptotically $\mathcal{O}(p^3)$, with p the number of columns in the library, this is computationally less expensive than regression in the entire library.

Combinations of these changes, such as a simultaneous addition and deletion, are more challenging and will also be explored. This approach is known as *abrupt-SINDy*, and it is depicted schematically in Fig. 2.2.

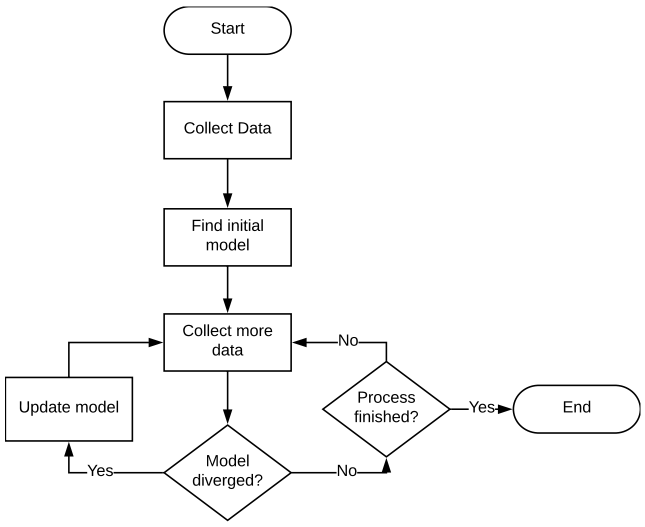


Figure 2.2: Adaptive SINDy flow chart. For an initial model and hyper parameter selection, a gridsearch is conducted. Next, we apply a predictor corrector scheme checking every t_{error} for model divergence using estimated Lyapunov time, and eventually update the model in a two step fashion.

2.3.1 *Baseline model*

First, we must identify a baseline SINDy model, and we use a gridsearch to determine the optimal hyper parameter selection. In gridsearch, all combinations of hyper parameters are tested and the best performing set is selected. This search is only performed once, locking in hyper parameters for future updates. The baseline model is characterized by the sparse coefficients in Ξ_0.

2.3.2 *Detecting model divergence*

It is essential to rapidly detect any change in the model, and we employ a classical predictor-corrector scheme [P12]. The predictor step is performed over a time τ_{pred}

in the interval $t, t + \tau_{\text{pred}}$ using the model valid at time t. The divergence of the predicted and measured state is computed at $t + \tau$ as $\|\Delta\mathbf{x}\| = \|\hat{\mathbf{x}}(t+\tau) - \mathbf{x}(t+\tau)\|$, where $\hat{\mathbf{x}}$ is the prediction and $\mathbf{x}$ is the measurement. The idea is to identify when the model and the measurement diverge faster than predicted by the dynamics of the system. For a chaotic system, the divergence of a trajectory is measured by the largest Lyapunov exponent of the system [P67], although a wealth of similar measures have been suggested [P68]. The Lyapunov exponent is defined as

$$\lambda = \lim_{\tau\to\infty} \lim_{\Delta\mathbf{x}(t_0)\to 0} \frac{\left\langle \log\left(\frac{\Delta\mathbf{x}(t_0+\tau)}{\Delta\mathbf{x}(t_0)}\right)\right\rangle}{\tau}, \tag{2.9}$$

and its inverse sets the fastest time scale. Here, the analogy of ensemble and time average is used, more precisely the local, finite-time equivalent [P69, P70]. An improvement can be achieved by exploiting an ensemble, e. g. by adding noise to the state $\mathbf{x}$ that corresponds to the given measurement accuracy. Since we know the dynamical system for the prediction step, the Lyapunov exponent is determined by evolving the tangent space with the system [P71, P72].

In our detection algorithm, we fix a fluctuation tolerance $\Delta\mathbf{x}$ and measure if the divergence time we find deviates from the expectation. If data are noisy, this tolerance must be significantly larger than the the typical fluctuation scale of the noise. Formally, the model and measurements diverge if the time-scale given by the local Lyapunov exponent and prediction horizon disagree. The local Lyapunov exponent is computed directly from the eigenvalues of the dynamical system [P71, P72]. The prediction horizon $T(t)$ is the first passage time where prediction $\hat{\mathbf{x}}(t + \Delta t)$ with initial condition $\mathbf{x}(t)$ and and measurement $\mathbf{x}(t+\Delta t)$ differ by more than $\Delta\mathbf{x}$:

$$T(t) = \underset{\Delta t}{\operatorname{argmax}} \|\hat{\mathbf{x}}(t+\Delta t) - \mathbf{x}(t+\Delta t)\| < \|\Delta\mathbf{x}\|. \tag{2.10}$$

Analogous to the local Lyapunov exponent, we compute the ratio $\log\|\Delta\mathbf{x}(t_0+\tau)/\Delta\mathbf{x}(t_0)\|$ as a measure for the divergence based on the measurement. For the model, we compute the local Lyapunov exponent as the average maximum eigenvalue $\bar{\lambda}(t) = \langle\lambda(t')\rangle_{t'\in[t,t+T]}$ with $\lambda(t) = \max(\lambda_i(t))$ and $\lambda_i v_i(t) = \partial f_j/\partial\mathbf{x}_k\big|_{\mathbf{x}(t)} v_i(t)$. Thus we compare the expected and observed trajectory divergence. Model and measurement have diverged at time t if the model time scale and the measured one differ:

$$\bar{\lambda}(t) > \alpha \frac{\log(\Delta\mathbf{x}) - \log(\bar{\Delta}(t))}{T(t)}. \tag{2.11}$$

If the model is not chaotic, but the measurement is chaotic, one must invert the inequality, as in Fig. 2.3. The empirical factor α accounts for finite-time statistics.

This method depends heavily on the particular system under investigation, including the dynamics, time scales, and sampling rate. In a practical implementation, these considerations must be handled carefully and automatically. It is important to note that we are able to formulate the divergence in terms of dynamical systems theory, because our model *is* a dynamical system, in other cases, such

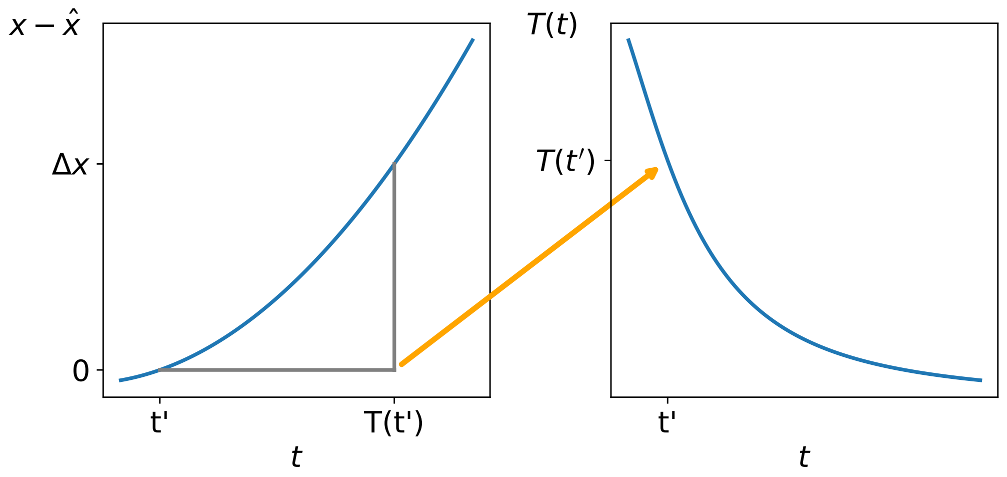


Figure 2.3: Sketch of the prediction horizon estimation. We use the observation $\mathbf{x}(t)$ as initial condition for the current model. Integration gives $\hat{\mathbf{x}}(t)$. The prediction horizon $T(t)$ is calculated according to Eq. (2.10). The prediction horizon is a function of time and the current model. It indicates divergence of model and observation. For details see text.

as artificial neural networks, this is not possible due to the limited mathematical framework.

2.3.3 *Adaptive model fitting*

After a change is detected, the following procedure is implemented to rapidly recover the model:

1. First, the new data is regressed onto the existing sparse structure Ξ_0 to identify varying parameters.
2. Next, we identify deletions of terms by performing the sparse regression on the sparse columns of $\mathbf{\Theta}$ that correspond to nonzero rows in Ξ_0. This is more efficient than identifying a new model, as we only seek to delete existing terms from the model.
3. Finally, if there is still a residual error, then a sparse model is fit for this error in the inactive columns of $\mathbf{\Theta}$ that correspond to zero rows in Ξ_0. In this way, new terms may be added to the model.

If the residual is sufficiently small after any step, the procedure ends. Alternatively, the procedure may be iterated until convergence. We are solving smaller regression problems by restricting our attention to subsets of the columns of $\mathbf{\Theta}$. These smaller regressions require less data and are less computationally expensive [P63], compared to fitting a new model. The deletion-addition procedure is performed after a model divergence is detected, using new transient data collected in an interval of size t_{update}.

2.4 RESULTS

In this section, we describe the results of the abrupt-SINDy framework on dynamical systems with abrupt changes, including parameter variation, deletion of terms, and addition of terms. The proposed algorithm is compared against the original SINDy algorithm, which is used to identify a new model from scratch, in terms of data required, computational time, and model accuracy.

In each case, we begin by running a gridsearch algorithm[P73][1] to identify the main parameters: α, the ridge regression regularization parameter; γ, the thresholding parameter; n_{degree}, the maximum degree of the polynomial feature transformation; and n_{fold}, the number of cross-validation runs. For scoring we use the explained variance score and conduct a five-fold cross validation for each point in the $(\alpha, \gamma, n_{\text{degree}})$ parameter grid. The parameters are listed in Table 2.1.

Parameter	Value
α	$0, 0.2, 0.4, 0.6, 0.8, 0.95$
γ	$0.1, 0.2, 0.4$
n_{degree}	$2, 3$
n_{fold}	5
Seed	42
CV	k-fold
Score	explained variance score

Table 2.1: Parameters for the grid search.

2.4.1 *Lorenz system*

The Lorenz system is a well-studied, and highly simplified, conceptual model for atmospheric circulation [P74]:

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - xz - y \\ \dot{z} &= xy - \beta z\end{aligned} \tag{2.12}$$

where the parameter ρ represents the heating of the atmosphere, corresponding to the Rayleigh number, σ corresponds to Prandtl number, and β to the aspect ratio [P75]. The parameters are set to $\rho = 28$, $\beta = 8/3$, $\sigma = 10$.

In the following we integrate the system numerically to produce a reference data set. We deliberately change the parameter ρ at $t = 40$ to $\rho = 15$ and at $t = 80$ back to $\rho = 28$, as shown in Fig. 2.4 and Fig. 2.5. These parametric changes lead to a bifurcation in the dynamics, and they are detected quickly. The subsequent adapted parameters are accurately detected up to two digits, as shown in Table 2.2. Because we are identifying the sparse model structure on a normalized library $\boldsymbol{\Theta}$, with zero mean and unit variance, we must de-bias the parameter estimates by

1 The user manual is located at `http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`.

computing a least-squares regression onto the sparse model structure in the original unnormalized variables. Otherwise, computing the least-squares regression in the normalized library, as is typically recommended in machine learning, would result in non-physical constant terms in the original unnormalized coordinates.

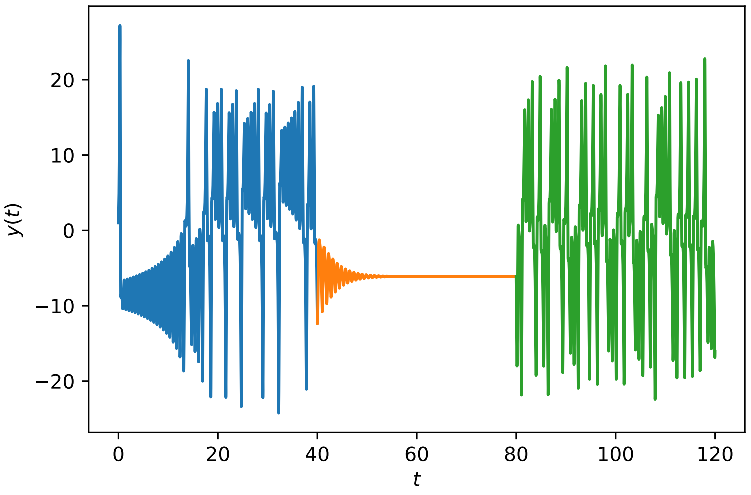


Figure 2.4: Time-series of the y coordinate of the Lorenz system. The blue and green segments correspond to the system parameters $\sigma = 10, \rho = 28, \beta = \frac{8}{3}$. The orange segment from $t = 40$ and $t = 80$ corresponds to the modified parameter $\rho = 15$. The initial condition is $\mathbf{x}_0 = (1, 1, 1)$.

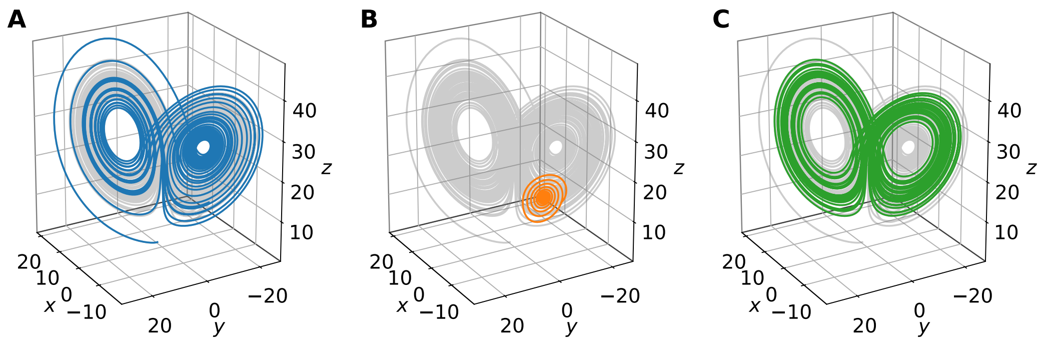


Figure 2.5: Lorenz system: Colors and parameters as in Fig. 2.4. In **A**, **B**, and **C** we show the first, second, and third segments of the trajectory in color with the concatenated trajectory in grey. The system changes from a butterfly attractor to a stable fixed point and back to a butterfly attractor.

Abrupt changes to the system parameters are detected using the prediction horizon from Eq. (2.10). When the system changes, the prediction horizon of the system should decrease, with smaller horizon corresponding to a more serious change. Conversely, the inverse time, corresponding to the Lyapunov exponent, should diverge. Fig. 2.6 exhibits this expected behavior. After a change is detected the model is rapidly recovered as shown in Table 2.2. It is important to confirm that the updated model accurately represents the structure of the true dynamics. Fig. 2.6 shows the norm of the model coefficients, $\|\xi - \hat{\xi}\|$, which is a measure of the distance between the estimated and true systems. Except for a short time ($t_{\text{update}} = 1$) after the abrupt change, the identified model closely agrees with the true model.

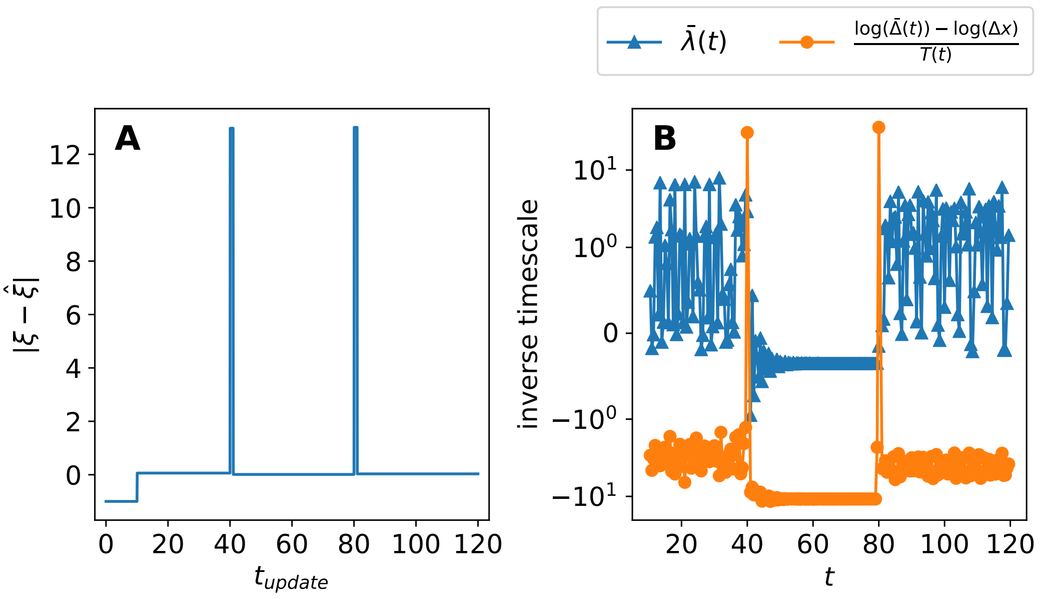


Figure 2.6: Lorenz system: **A** We show the model accuracy over time. For coefficients, see Table 2.2. For $t \leq 10$ no model is available and $\|\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}\| = -1$. At both switch points, $t = 40$ and $t = 80$, $t_{\text{update}} = 1$ is needed to update the model. During this interval, a fallback solution, e. g. DMD could be implemented. Note that the accuracy metric requires knowledge about the ground truth and thus is only available in a hindcast scenario. **B** Evaluation of Eq. (2.10). At both switch points, $t = 40$ and $t = 80$, we quickly detect the divergence of model and measurement. The parameters are $t_{\text{model}} = 10, t_{\text{update}} = 1, t_{\text{error}} = 0.5$, and $\Delta\mathbf{x} = 1.0$.

t_{detected}	t_{update}	Equations
0.00	10.0	$\dot{x} = -10.0x + 10.0y$ $\dot{y} = 27.96x - 0.99y - 1.0xz$ $\dot{z} = -2.67z + 1.0xy$
40.01	41.0	$\dot{x} = -10.0x + 10.0y$ $\dot{y} = 15.0x - 1.0y - 1.0xz$ $\dot{z} = -2.67z + 1.0xy$
80.02	81.0	$\dot{x} = -10.0x + 10.0y$ $\dot{y} = 27.98x - 1.0y - 1.0xz$ $\dot{z} = -2.67z + 1.0xy$

Table 2.2: Lorenz system: detection and update times, along with identified equations. The detection time coincides up to the second digit with the true switching time. The rapidly identified model agrees well with the true model structure and parameters. Coefficients are rounded to the second digit.

Effects of noise and data volume

An important set of practical considerations include how noise and the amount of data influence the speed of change detection and the accuracy of subsequent model recovery. Both the noise robustness and the amount of data required will change for a new problem, and here we report trends for this specific case. In addition, the amount of data required is also related to the sampling rate, which is the subject of ongoing investigation; in some cases, higher sampling time may even degrade model performance due to numerical effects [P58].

Fig. 2.7 shows the model fit following the abrupt change, comparing both the abrupt-SINDy method, which uses information about the existing model structure, and the standard SINDy method, which re-identifies the model from scratch following a detected change. In this figure, the model quality is shown as a function of the amount of data collected after the change.

The abrupt-SINDy model is able to identify more accurate models in a very short amount of time, given by $t_{\text{update}} \approx 0.1$. At this point, the standard SINDy method shows comparable error, however for even smaller times, the data are no longer sufficient for the conventional method. Since the adaptive method starts near the optimal solution, larger data sets do not degrade the model, which was an unexpected additional advantage.

Fig. 2.8 explores the effect of additive noise on the derivative on the abrupt-SINDy and standard SINDy algorithms. Note that in practice noise will typically be added to the measurement of $\mathbf{x}$, as in the original SINDy algorithm [P40], requiring a denoising derivative [P58, P59]; however, simple additive noise on the derivative is useful to investigate the robustness of the regression procedure. Abrupt-SINDy has considerably higher noise tolerance than the standard algorithm, as it must identify fewer unknown coefficients. In fact, it is able to handle approximately an order of magnitude more noise before failing to identify a model. Generally, increasing the volume of data collection improves the model. The critical point in the abrupt-SINDy curves corresponds to when small but dynamically important terms are mis-identified as a result of insufficient signal-to-noise. Although the noise and chaotic signal cannot be easily distinguished for small signal-to-noise, it may be possible to distinguish between them using a spectral analysis, since chaos yields red noise in contrast to the white additive noise.

2.4.2 *Van der Pol*

As a second example, we consider the famous nonlinear Van der Pol oscillator [P76]. We include additional quadratic nonlinearities αx^2 and αy^2 to study the ability of our method to capture structural changes when these terms are added and removed abruptly. This example focuses on the important class of periodic phenomena, in contrast to the chaotic Lorenz dynamics. The modified Van der Pol oscillator is described by the following equations:

$$\begin{aligned} \dot{x} &= y - \alpha y^2 \\ \dot{y} &= \mu(1 - x^2)y - x + \alpha x^2 , \end{aligned} \tag{2.13}$$

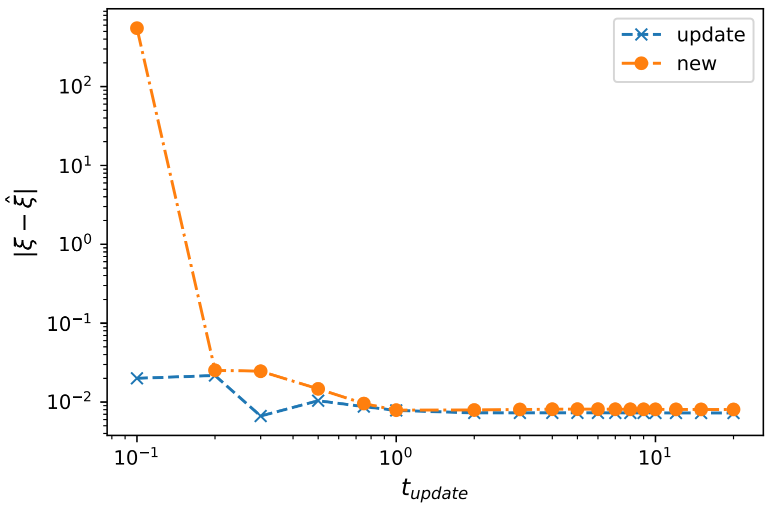


Figure 2.7: Lorenz system: We show the model accuracy versus the amount of data used to update (blue $\times$) or re-fit (orange dot) respectively. Data are collected in the interval $[40, 40 + t_{\text{update}}]$ just after the first change of the system dynamics. The number of data points for $t_{\text{update}} = 0.1$ are $N = 25$, for $t_{\text{update}} = 10$ we have 2500 points. At $t_{\text{update}} \simeq 1$, updating and re-fitting methods become comparable. However, for smaller update times, or less data, respectively, the fraction of transient data becomes too small for identifying the exact model from scratch. Updating the model needs less data for the same accuracy or achieves higher accuracy with the same amount of data.

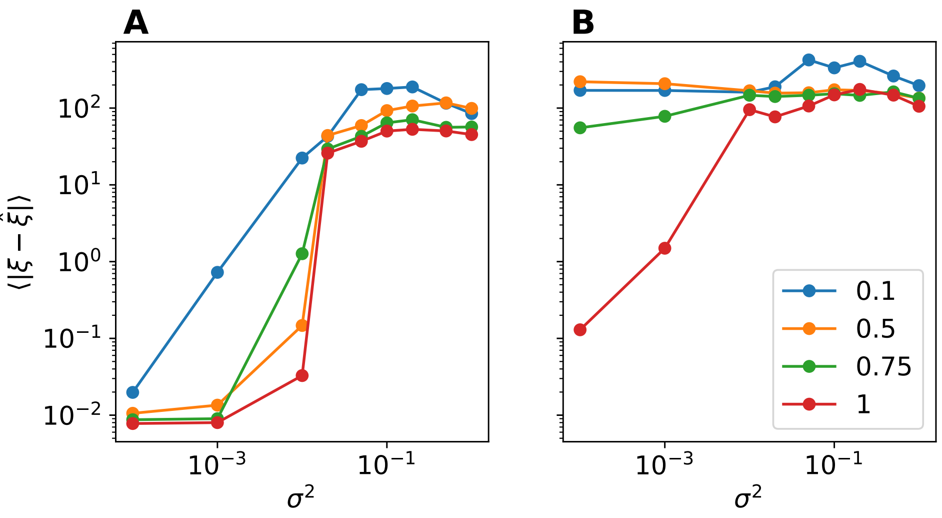


Figure 2.8: Lorenz system: We show the noise robustness of model accuracy. In A) we use the previous knowledge and update the model; in B) we make a new fit only re-using the previously discovered hyper-parameters. The curves are parametrized by t_{update}, *cf.* Fig. 2.7. The accuracy measure is very noise sensitive, as distinction between library functions gets lost. At a signal to noise ratio of approximately 1, no accurate model can be obtained with either model. At lower noise ratios, updating the model achieves higher accuracy (the library is smaller). In both cases, accuracy scales approximately logarithmically with t_{update}.

where $\mu > 0$ is a parameter controlling the nonlinear damping, and α parameterizes the additional quadratic nonlinearity. The reference data set is shown in Fig. 2.9, with $\mu = 7.5$ and $\alpha = 0$ for $t \in [0, 100]$, which results in a canonical periodic orbit. At $t = 100$ we introduce a structural change, switching on the quadratic nonlinearity ($\alpha = -0.25$), and driving the system to a stable fixed point. We also modify the parameter μ, setting it to $\mu = 6.0$. Finally, at $t = 200$, we switch off the additional nonlinearity ($\alpha = 0$) and keep $\mu = 6$.

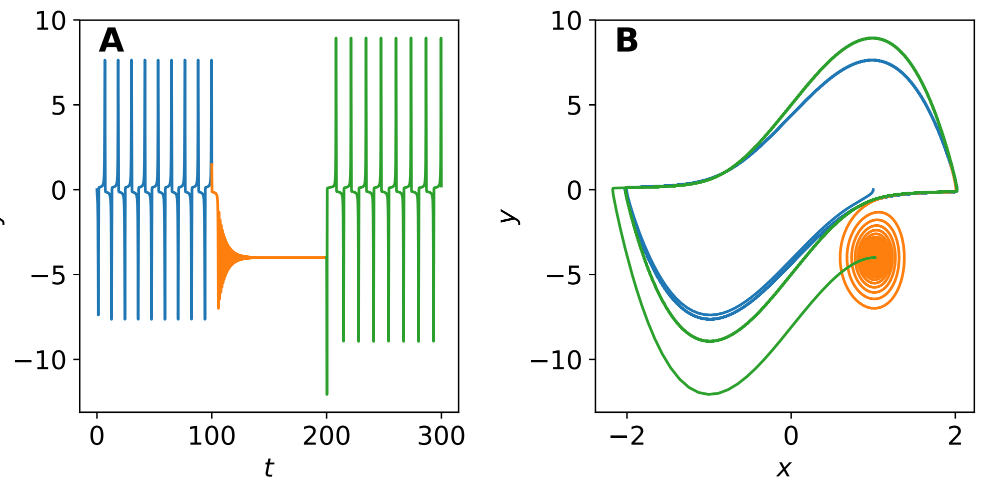


Figure 2.9: Van der Pol system with parameters $\mu = 5, \alpha = 0$ (blue), $\mu = 7.5, \alpha = -0.25$ (orange), $\mu = 6.0, \alpha = 0$ (green). **A**: time evolution of the y-coordinate. **B** phase-space-trajectory x, y.

Table 2.3 shows the corresponding models recovered using the abrupt-SINDy method. The change is detected using the Lyapunov time defined in Eq. (2.11), as shown in Fig. 2.10. Again, the estimated Lyapunov time (Fig. 2.10 **B**) captures the the changes in the model, which correspond to peaks in structural model error (Fig. 2.10 **A**). While the first and third stage are indeed identified correctly, the term $-1.25x$ is preferred over $-x - 0.25x^2$ in the sparse estimate for $\dot{y}$ in the orange trajectory. However, since both terms look similar near the fixed point at $x \sim 1$, this describes the dynamics well. This type of mis-identification often occurs in data mining when features are highly correlated [P63] and is more related to sparse regression in general than the proposed abrupt-SINDy. For dynamic system identification, the correct nonlinearity could be resolved by obtaining more transient data, i.e. by perturbing the system through actuation. However, this model may be sufficient for control while a more accurate model is identified.

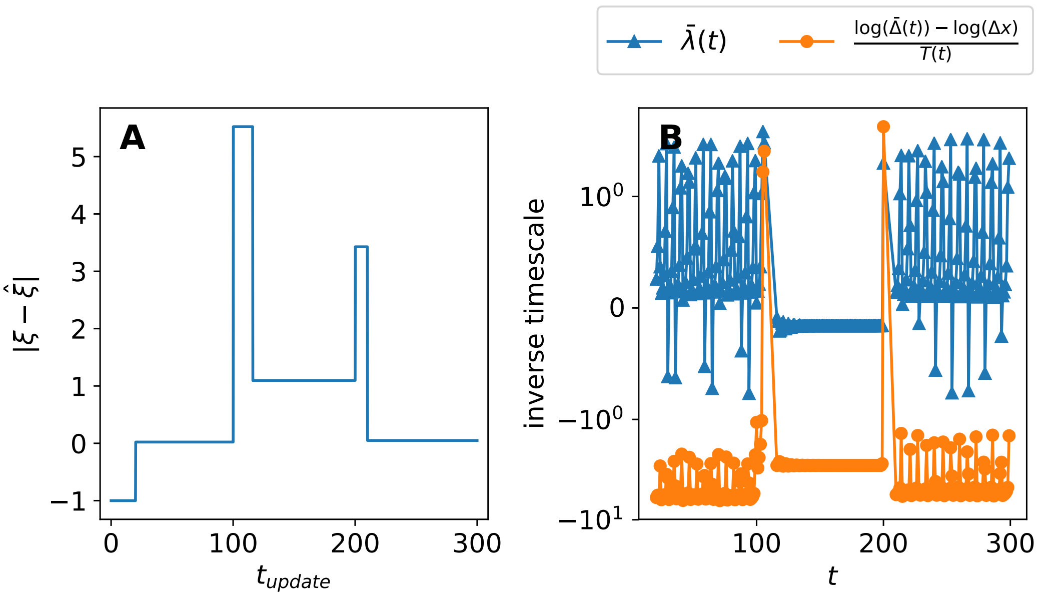


Figure 2.10: Van der Pol system: Evaluation of Eq. (2.10). Parameters: $t_{\text{model}} = 20, t_{\text{update}} = 10, t_{\text{error}} = 1, \Delta x = 1.5$.

t_{detected}	t_{update}	Equations
0.00	20.01	$\dot{x} = 1.0y$ $\dot{y} = -1.0x + 4.99y - 4.99x^2y$
106.39	116.00	$\dot{x} = 0.99y + 0.25y^2$ $\dot{y} = -1.26x + 7.46y - 7.46x^2y$
200.12	210.00	$\dot{x} = 1.0y$ $\dot{y} = -1.0x + 5.98y - 5.98x^2y$

Table 2.3: Van der Pol system: Summary of the discovered equations. Coefficients are rounded to the second digit.

2.5 CONCLUSIONS

In this work, we develop an adaptive nonlinear system identification strategy designed to rapidly recover nonlinear models from limited data following an abrupt change to the system dynamics. The sparse identification of nonlinear dynamics (SINDy) framework is ideal for change detection and model recovery, as it relies on parsimony to select the fewest active terms required to model the dynamics. In our adaptive abrupt-SINDy method, we rely on previously identified models to identify the fewest *changes* required to recover the model. This modified algorithm is shown to be highly effective at model recovery following an abrupt change, requiring less data, less computation time, and having improved noise robustness over identifying a new model from scratch. The abrupt-SINDy method is demonstrated on several numerical examples exhibiting chaotic dynamics and periodic dynamics, as well as parametric and structural changes, enabling real-time model recovery.

There are limitations of the method which can be addressed by several promising directions that may be pursued to improve the abrupt-SINDy method:

1. **Fallback models:** In the current implementation, after a change has been detected, the old model will be used until enough data is collected to identify a new model. The dynamic mode decomposition [P17] provides an alternative fallback model, that may be identified rapidly with even less data. Additionally, instead of relying on a sparse update to the current model, it is sensible to also maintain a library of past models for rapid characterization [P77].

2. **Hyperparameterization:** In the initial prototype, the hyper-parameters $\Delta_\mathbf{x}$ and t_{update} are fixed. Over time, an improved algorithm may learn and adapt optimal hyper-parameters.

3. **Comprehensive Lyapunov time estimation:** According to Eq. (2.10), the Lyapunov time $T(t|\Delta\mathbf{x})$ is estimated for a fixed $\Delta\mathbf{x}$. Estimating the time for a range of values, i.e. $\Delta\mathbf{x} \in (0, \Delta\mathbf{x}_{\text{max}}]$, will be more robust and may provide a richer analysis without requiring additional data. Further investigation must be made into the case of chaotic systems, where the numerical calculation of the Lyapunov exponent may fail to reveal divergence due to the fact of simple averaging over time. Because of the importance of the detection of model divergence, this is a particularly important area of future research.

4. **Advanced optimization and objectives:** Looking forward, advanced optimization techniques may be used to further improve the adaptation to system changes. Depending on the system, other objectives may be optimized, either by including regularization or in a multi-objective optimization.

The proposed abrupt-SINDy framework is promising for the real-time recovery of nonlinear models following abrupt changes. It will be interesting to compare with other recent algorithms that learn local dynamics for control in response to abrupt changes [P78]. Future work will be required to demonstrate this method on more sophisticated engineering problems and to incorporate it in controllers. The abrupt-SINDy modeling framework may also help inform current rapid learning strategies in neural network architectures [P2, P3, P4], potentially allowing dynamical systems methods to inform rapid training paradigms in deep learning.

ACKNOWLEDGEMENTS

MQ was supported by a fellowship within the FITweltweit program of the German Academic Exchange Service (DAAD). MQ and MA acknowledge support by the European Erasmus SME/HPC project (588372-EPP-1-2017-1-IE-EPPKA2-KA). SLB acknowledges support from the ARO and AFOSR Young Investigator Programs (ARO W911NF-17-1-0422 and AFOSR FA9550-18-1-0200). SLB and JNK acknowledge support from DARPA (HR0011-16-C-0016). We would like to thank the anonymous reviewers for their comments which helped to improve this manuscript. We also acknowledge valuable discussions related to abrupt model recovery and programming wisdom with Dennis Bernstein, Karthik Duraisamy, Thomas Isele, Eurika Kaiser, and Hod Lipson. All code for this work is publicly available on Github: `https://github.com/0hjeah/sparsereg`.

REFERENCES

[P1] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. Vol. 1. MIT press Cambridge, 2016.

[P2] Li Fei-Fei, Rob Fergus, and Pietro Perona. "One-shot learning of object categories." In: *IEEE Trans. Pattern Anal. Machine Intell.* 28.4 (Apr. 2006), pp. 594–611. DOI: 10.1109/tpami.2006.79.

[P3] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. "Matching networks for one shot learning." In: *Advances in Neural Information Processing Systems*. 2016, pp. 3630–3638.

[P4] Charles B Delahunt and J Nathan Kutz. *Olfactory*. 2018. DOI: 10.5040/9781408173503.00000041.

[P5] Sinno Jialin Pan and Qiang Yang. "A Survey on Transfer Learning." In: *IEEE Trans. Knowl. Data Eng.* 22.10 (Oct. 2010), pp. 1345–1359. DOI: 10.1109/tkde.2009.191.

[P6] Steven L. Brunton and Bernd R. Noack. "Closed-Loop Turbulence Control: Progress and Challenges." In: *Appl. Mech. Rev* 67.5 (Aug. 2015), p. 050801. DOI: 10.1115/1.4031175.

[P7] Catharine H Rankin. "Invertebrate Learning: What Can't a Worm Learn?" In: *Curr. Biol.* 14.15 (Aug. 2004), R617–R618. DOI: 10.1016/j.cub.2004.07.044.

[P8] J. R. Whitlock. "Learning Induces Long-Term Potentiation in the Hippocampus." In: *Science* 313.5790 (Aug. 2006), pp. 1093–1097. DOI: 10.1126/science.1128134.

[P9] Joshua P. Johansen, Christopher K. Cain, Linnaea E. Ostroff, and Joseph E. LeDoux. "Molecular Mechanisms of Fear Learning and Memory." In: *Cell* 147.3 (Oct. 2011), pp. 509–524. DOI: 10.1016/j.cell.2011.10.009.

[P10] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems." In: *J. Basic Engineering* 82.1 (1960), p. 35. DOI: 10.1115/1.3662552.

[P11] B. L. Ho and R. E. Kalman. "Effective Construction of Linear State-Variable Models from Input/Output Data." In: *Proc. of the 3rd Ann. Allerton Conference on Circuit and System Theory*. 1965, pp. 449–459.

[P12] N.A. Gershenfeld. *The Nature of Mathematical Modeling*. Cambridge University Press, 1999.

[P13] J.-N. Juang and R. S. Pappa. "An eigensystem realization algorithm for modal parameter identification and model reduction." In: *Journal of Guidance, Control, and Dynamics* 8.5 (Sept. 1985), pp. 620–627. DOI: 10.2514/3.20031.

[P14] Peter J. Schmid. "Dynamic mode decomposition of numerical and experimental data." English. In: *J. Fluid Mech.* 656 (July 2010), pp. 5–28. DOI: 10.1017/s0022112010001217.

[P15] Clarence W. Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S. Henningson. "Spectral analysis of nonlinear flows." In: *J. Fluid Mech.* 641 (Nov. 2009), p. 115. DOI: 10.1017/s0022112009992059.

[P16] J. Nathan Kutz, Steven L. Brunton, Dirk M. Luchtenburg, Clarence W. Rowley, and Jonathan H. Tu. "On dynamic mode decomposition: Theory and applications." In: *JCD* 1.2 (Dec. 2014), pp. 391–421. DOI: 10.3934/jcd.2014.1.391.

[P17] J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, and Joshua L. Proctor. *Dynamic Mode Decomposition. Data-Driven Modeling of Complex Systems*. Society for Industrial and Applied Mathematics, Nov. 2016. DOI: 10.1137/1.9781611974508.

[P18] Hirotugu Akaike. "Fitting autoregressive models for prediction." In: *Ann Inst Stat Math* 21.1 (Dec. 1969), pp. 243–247. DOI: 10.1007/bf02532251.

[P19] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer Texts in Statistics. Springer New York, 1996. DOI: 10.1007/978-1-4757-2526-1.

[P20] R. González-García, R. Rico-Martínez, and I.G. Kevrekidis. "Identification of distributed parameter systems: A neural net based approach." In: *Computers & Chemical Engineering* 22 (Mar. 1998), S965–S968. DOI: 10.1016/s0098-1354(98)00191-4.

[P21] Enoch Yeung, Soumya Kundu, and Nathan Hodas. *Learning Deep Neural Network Representations for Koopman Operators of Nonlinear Dynamical Systems*. 2017. eprint: 1708.06850 (cs.LG).

[P22] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. "Learning Koopman Invariant Subspaces for Dynamic Mode Decomposition." In: *Advances in Neural Information Processing Systems*. 2017, pp. 1130–1140.

[P23] Christoph Wehmeyer and Frank Noé. "Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics." In: *The Journal of Chemical Physics* 148.24 (June 2018), p. 241703. DOI: 10.1063/1.5011399.

[P24] Andreas Mardt, Luca Pasquali, Hao Wu, and Frank Noé. "VAMPnets for deep learning of molecular kinetics." In: *Nat Commun* 9.1 (Jan. 2018). DOI: 10.1038/s41467-017-02388-1.

[P25] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. "Deep learning for universal linear embeddings of nonlinear dynamics." In: *Proc. Natl. Acad. Sci. U.S.A.* (2017). arXiv: 1712.09707 [math.DS]. Submitted.

[P26] B. O. Koopman. "Hamiltonian Systems and Transformation in Hilbert Space." In: *Proceedings of the National Academy of Sciences* 17.5 (May 1931), pp. 315–318. DOI: 10.1073/pnas.17.5.315.

[P27] Igor Mezić. "Spectral Properties of Dynamical Systems, Model Reduction and Decompositions." In: *Nonlinear Dyn* 41.1-3 (Aug. 2005), pp. 309–325. DOI: 10.1007/s11071-005-2824-x.

[P28] Mantas Lukoševičius and Herbert Jaeger. "Reservoir computing approaches to recurrent neural network training." In: *Computer Science Review* 3.3 (Aug. 2009), pp. 127–149. DOI: 10.1016/j.cosrev.2009.03.005.

[P29] Zhixin Lu, Jaideep Pathak, Brian Hunt, Michelle Girvan, Roger Brockett, and Edward Ott. "Reservoir observers: Model-free inference of unmeasured variables in chaotic systems." In: *Chaos* 27.4 (Apr. 2017), p. 041102. DOI: 10.1063/1.4979665.

[P30] G.B. Dantzig, R. Cottle, and Y.P. Aneja. *Mathematical programming: Essays in honor of George B. Dantzig*. Mathematical Programming: Essays in Honor of George B. Dantzig Bd. 1. North-Holland, 1985.

[P31] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[P32] J. Bongard and H. Lipson. "Automated reverse engineering of nonlinear dynamical systems." In: *Proceedings of the National Academy of Sciences* 104.24 (June 2007), pp. 9943–9948. DOI: 10.1073/pnas.0609476104.

[P33] Michael Schmidt and Hod Lipson. "Distilling Free-Form Natural Laws from Experimental Data." In: *Science* 324.5923 (Apr. 2009), pp. 81–85. DOI: 10.1126/science.1165893.

[P34] Michael D Schmidt, Ravishankar R Vallabhajosyula, Jerry W Jenkins, Jonathan E Hood, Abhishek S Soni, John P Wikswo, and Hod Lipson. "Automated refinement and inference of analytical models for metabolic networks." In: *Phys. Biol.* 8.5 (Aug. 2011), p. 055011. DOI: 10.1088/1478-3975/8/5/055011.

[P35] William G. La Cava and Kourosh Danai. "Gradient-based adaptation of continuous dynamic model structures." In: *Int. J. Syst. Sci.* 47.1 (Aug. 2015), pp. 249–263. DOI: 10.1080/00207721.2015.1069905.

[P36] William La Cava, Kourosh Danai, Lee Spector, Paul Fleming, Alan Wright, and Matthew Lackner. "Automatic identification of wind turbine models using evolutionary multiobjective optimization." In: *Renewable Energy* 87.October 2015 (Mar. 2016), pp. 892–902. DOI: 10.1016/j.renene.2015.09.068.

[P37] Markus Quade, Markus Abel, Kamran Shafi, Robert K. Niven, and Bernd R. Noack. "Prediction of dynamical systems by symbolic regression." In: *Phys. Rev. E* 94.1 (July 2016), p. 012214. DOI: 10.1103/physreve.94.012214.

[P38] Julien Gout, Markus Quade, Kamran Shafi, Robert K. Niven, and Markus Abel. "Synchronization control of oscillator networks using symbolic regression." In: *Nonlinear Dyn* 91.2 (Nov. 2017), pp. 1001–1021. DOI: 10.1007/s11071-017-3925-z.

[P39] Thomas Duriez, Steven L. Brunton, and Bernd R. Noack. *Machine Learning Control – Taming Nonlinear Dynamics and Turbulence*. Vol. 116. Fluid Mechanics and Its Applications. Springer International Publishing, 2017. DOI: 10.1007/978-3-319-40624-4.

[P40] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems." In: *Proc Natl Acad Sci USA* 113.15 (Mar. 2016), pp. 3932–3937. DOI: 10.1073/pnas.1517384113.

[P41] H. Voss, M. Bünner, and M. Abel. "Identification of continuous, spatiotemporal systems." In: *Phys. Rev. E* 57.3 (3 Mar. 1998), pp. 2820–2823. DOI: 10.1103/physreve.57.2820.

[P42] Trent McConaghy. "FFX: Fast, Scalable, Deterministic Symbolic Regression Technology." In: *Genetic and Evolutionary Computation*. Springer New York, 2011, pp. 235–260. DOI: 10.1007/978-1-4614-1770-5_13.

[P43] Markus Abel, Karsten Ahnert, Jürgen Kurths, and Simon Mandelj. "Additive nonparametric reconstruction of dynamical systems from time series." In: *Phys. Rev. E* 71.1 (Jan. 2005), p. 015203. DOI: 10.1103/physreve.71.015203.

[P44] Henning U. Voss, Paul Kolodner, Markus Abel, and Jürgen Kurths. "Amplitude Equations from Spatiotemporal Binary-Fluid Convection Data." In: *Phys. Rev. Lett.* 83.17 (17 Oct. 1999), pp. 3422–3425. DOI: 10.1103/physrevlett.83.3422.

[P45] Markus Abel. "Nonparametric Modeling And Spatiotemporal Dynamical Systems." In: *Int. J. Bifurcation Chaos* 14.06 (June 2004), pp. 2027–2039. DOI: 10.1142/s0218127404010382.

[P46] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. *Sparse identification of nonlinear dynamics for model predictive control in the low-data limit*. 2017. arXiv: 1711.05501 [math.OC].

[P47] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Data-driven discovery of partial differential equations." In: *Sci. Adv.* 3.4 (Apr. 2017), e1602614. DOI: 10.1126/sciadv.1602614.

[P48] Hayden Schaeffer. "Learning partial differential equations via data discovery and sparse optimization." In: *Proc. R. Soc. A*. Vol. 473. 2197. The Royal Society. 2017, p. 20160446.

[P49] Jean-Christophe Loiseau and Steven L. Brunton. "Constrained sparse Galerkin regression." In: *J. Fluid Mech.* 838 (Jan. 2018), pp. 42–67. DOI: 10.1017/jfm.2017.823.

[P50] Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, Eurika Kaiser, and J. Nathan Kutz. "Chaos as an intermittently forced linear system." In: *Nat Commun* 8.1 (May 2017). DOI: 10.1038/s41467-017-00030-8.

[P51] Giang Tran and Rachel Ward. "Exact Recovery of Chaotic Systems from Highly Corrupted Data." In: *Multiscale Model. Simul.* 15.3 (Jan. 2017), pp. 1108–1129. DOI: 10.1137/16m1086637.

[P52] Hayden Schaeffer and Scott G. McCalla. "Sparse model selection via integral terms." In: *Phys. Rev. E* 96.2 (Aug. 2017), p. 023302. DOI: 10.1103/physreve.96.023302.

[P53] S. L. Brunton, J. L. Proctor, and J. N. Kutz. "Sparse Identification of Nonlinear Dynamics with Control (SINDYc)." In: *IFAC NOLCOS* 49.18 (2016), pp. 710–715.

[P54] N. M. Mangan, J. N. Kutz, S. L. Brunton, and J. L. Proctor. "Model selection for dynamical systems via sparse regression and information criteria." In: *Proc. R. Soc. A* 473.2204 (Aug. 2017), p. 20170009. DOI: 10.1098/rspa.2017.0009.

[P55] Robert Tibshirani. "Regression shrinkage and selection via the lasso: A retrospective. Regression Shrinkage and Selection via the Lasso." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.3 (Apr. 2011), pp. 273–282. DOI: 10.1111/j.1467-9868.2011.00771.x.

[P56] Trevor Hastie, Jerome Friedman, and Robert Tibshirani. *The Elements of Statistical Learning*. Vol. 2. 1. Springer New York, 2001. DOI: 10.1007/978-0-387-21606-5.

[P57] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer New York, 2013. DOI: 10.1007/978-1-4614-7138-7.

[P58] Karsten Ahnert and Markus Abel. "Numerical differentiation of experimental data: Local versus global methods." In: *Comput. Phys. Commun.* 177.10 (Nov. 2007), pp. 764–774. DOI: 10.1016/j.cpc.2007.03.009.

[P59] Rick Chartrand. "Numerical Differentiation of Noisy, Nonsmooth Data." In: *ISRN Applied Mathematics* 2011 (2011), pp. 1–11. DOI: 10.5402/2011/164564.

[P60] Wen-Xu Wang, Rui Yang, Ying-Cheng Lai, Vassilios Kovanis, and Celso Grebogi. "Predicting Catastrophes in Nonlinear Dynamical Systems by Compressive Sensing." In: *Phys. Rev. Lett.* 106.15 (Apr. 2011), pp. 154101-1–154101-4. DOI: 10.1103/physrevlett.106.154101.

[P61] D.L. Donoho. "Compressed sensing." In: *IEEE Trans. Inform. Theory* 52.4 (Apr. 2006), pp. 1289–1306. DOI: 10.1109/tit.2006.871582.

[P62] E. J. Candès. "Compressive Sensing." In: *Proceedings of the International Congress of Mathematics* (2006).

[P63] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. "Feature Selection. A Data Perspective." In: *CSUR* 50.6 (Dec. 2017), pp. 1–45. DOI: 10.1145/3136625.

[P64] Linan Zhang and Hayden Schaeffer. *On the Convergence of the SINDy Algorithm*. May 2018. arXiv: 1805.06445 [math.OC].

[P65] Krithika Manohar, Bingni W. Brunton, J. Nathan Kutz, and Steven L. Brunton. "Data-Driven Sparse Sensor Placement." In: *IEEE Control Syst. Mag.* (2017). arXiv: 1701.07569 [math.OC]. Submitted.

[P66] N. Benjamin Erichson, Ariana Mendible, Sophie Wihlborn, and J. Nathan Kutz. "Randomized nonnegative matrix factorization." In: *Pattern Recognit. Lett.* 104 (Mar. 2018), pp. 1–7. DOI: 10.1016/j.patrec.2018.01.007.

[P67] Edward Ott. *Chaos in Dynamical Systems*. Cambridge University Press, 2002. DOI: 10.1017/cbo9780511803260.

[P68] *Handbook of Chaos Control*. Wiley-VCH Verlag GmbH & Co. KGaA, Oct. 2007. DOI: 10.1002/9783527622313.

[P69] Ying-Cheng Lai and Tamás Tél. *Transient Chaos*. Vol. 173. Springer New York, 2011. DOI: 10.1007/978-1-4419-6987-3.

[P70] Ruiqiang Ding and Jianping Li. "Nonlinear finite-time Lyapunov exponent and predictability." In: *Phys. Lett. A* 364.5 (May 2007), pp. 396–400. DOI: 10.1016/j.physleta.2006.11.094.

[P71] Holger Kantz. "A robust method to estimate the maximal Lyapunov exponent of a time series." In: *Phys. Lett. A* 185.1 (Jan. 1994), pp. 77–87. DOI: 10.1016/0375-9601(94)90991-1.

[P72] Arkady Pikovsky and Antonio Politi. "Dynamic localization of Lyapunov vectors in spacetime chaos." In: *Nonlinearity* 11.4 (July 1998), pp. 1049–1062. DOI: 10.1088/0951-7715/11/4/016.

[P73] G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, and A. Mueller. "Scikit-learn. Machine Learning Without Learning the Machinery." In: *GetMobile: Mobile Comp. and Comm.* 19.1 (June 2015), pp. 29–33. DOI: 10.1145/2786984.2786995.

[P74] Edward N. Lorenz. "Deterministic Nonperiodic Flow." In: *J. Atmos. Sci.* 20.2 (Mar. 1963), pp. 130–141. DOI: 10.1175/1520-0469(1963)020<0130:dnf>2.0.co;2.

[P75] Steven H Strogatz. *Nonlinear dynamics and chaos: With applications to physics, biology, chemistry, and engineering*. Westview press, 2014.

[P76] Balthasar Van der Pol. "A theory of the amplitude of free and forced triode vibrations." In: *Radio Review* 1.1920 (1920), pp. 701–710.

[P77] Steven L. Brunton, Jonathan H. Tu, Ido Bright, and J. Nathan Kutz. "Compressive Sensing and Low-Rank Libraries for Classification of Bifurcation Regimes in Nonlinear Dynamical Systems." In: *SIAM J. Appl. Dyn. Syst.* 13.4 (Jan. 2014), pp. 1716–1732. DOI: 10.1137/130949282.

[P78] Melkior Ornik, Arie Israel, and Ufuk Topcu. *Control-Oriented Learning on the Fly*. 2017. arXiv: 1709.04889 [math.OC].

3 PREDICTION OF DYNAMICAL SYSTEMS BY SYMBOLIC REGRESSION

by Markus Quade[1, 2], Markus Abel[1, 2], Kamran Shafi[3], Bernd R. Noack[4, 5], Robert K. Niven[3]

[1] Universität Potsdam, Institut für Physik und Astronomie, Karl-Liebknecht-Straße 24/25, 14476 Potsdam, Germany
[2] Ambrosys GmbH, David-Gilly Straße 1, 14469 Potsdam, Germany
[3] School of Engineering and Information Technology, The University of New South Wales, Canberra ACT 2600, Australia
[4] Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur LIMSI-CNRS, BP 133, 91403 Orsay cedex, France
[5] Institut für Strömungsmechanik, Technische Universität Braunschweig, Hermann-Blenk-Straße 37, 38108 Braunschweig, Germany

This paper has been publish in Physical Review E. We investigate the prediction of dynamical systems by symbolic regression applying both sparse regression and genetic programming based symbolic regression. Three examples are used to showcase the methods: the harmonic oscillator, coupled FitzHugh-Nagumo oscillators and solar power production data from a local solar panel.

Abstract

We study the modeling and prediction of dynamical systems based on conventional models derived from measurements. Such algorithms are highly desirable in situations where the underlying dynamics are hard to model from physical principles or simplified models need to be found. We focus on symbolic regression methods as a part of machine learning. These algorithms are capable of learning an analytically tractable model from data, a highly valuable property. Symbolic regression methods can be considered as generalized regression methods. We investigate two particular algorithms, the so-called fast function extraction which is a generalized linear regression algorithm, and genetic programming which is a very general method. Both are able to combine functions in a certain way such that a good model for the prediction of the temporal evolution of a dynamical system can be identified. We illustrate the algorithms by finding a prediction for the evolution of a harmonic oscillator based on measurements, by detecting an arriving front in an excitable system, and as a real-world application, the prediction of solar power production based on energy production observations at a given site together with the weather forecast.

3.1 INTRODUCTION

The prediction of the behavior of dynamical systems is of fundamental importance in all scientific disciplines. Since ancient times, philosophers and scientists have tried to formulate observational models and infer future states of such systems. Applications include topics as diverse as weather forecasting [Q1], the prediction of the motion of the planets [Q2], or the estimation of quantum evolution [Q3]. The common ingredient of such systems - at least in natural sciences - is the existence of an underlying mathematical model which can be applied as the predictor. In recent years, the use of Machine Learning (ML) methods -synonymously used here for artificial intelligence- complemented the formulation of such mathematical models through the application of advanced data analysis algorithms that allow accurate estimation of observed dynamics by learning automatically from the given observations and building models in terms of their own modelling languages. Artificial Neural Networkss (ANNs) are one example of such techniques that are popularly applied to model dynamic phenomena. ANNs are structured as networks of soft weights organized in layers or so-called neurons or hidden units. One problem of type approaches is the difficult-to-interpret black-box nature of the learnt models. Symbolic regression-based approaches, such as Genetic Programming (GP), provide alternative ML methods that are recently gaining increasing popularity. These methods, similar to other ML counterparts, learn models from observed data and act as good predictors of the future states of dynamical systems. Their added advantages over other methods include the interpretable nature of their learnt models and a flexible and weakly-typed [Q4] modelling language that allows them to be applied to a variety of domains and problems. We illustrate the hierarchy of models and their relation in Fig. 3.1.

Undoubtedly, the methods used most often in ML are neural networks. These algorithms are inspired by the architecture of the human brain, with several layers of neurons, as used in deep learning. In the present study, involving deterministic systems, we want to use a certain branch of ML, namely symbolic regression. This technique joins the classical, equation-oriented approach with its computer-scientific equivalent. In this publication we do not present any major improvements in the algorithms; rather we demonstrate how one can apply symbolic regression to identify and predict the future state of dynamical systems.

Symbolic regression algorithms work by exploring a function space, which is generally bounded by a preselected set of mathematical operators and operands (variables, constants, etc.), using a population of randomly generated candidate solutions. Each candidate solution encoded as a tree essentially works as a function and is evaluated based on its fitness or in other words its ability to match the observed output. These candidate solutions are evolved using a fitness-weighted selection mechanism and different recombination and variation operators. One common problem in symbolic regression is the bloating effect which is caused by excessive lengthening of individual solutions or filling of the population by large number of solutions with low fitness. In this work we use a multi-objective function evaluation mechanism to avoid this problem by including minimizing the solution length as an explicit objective in the fitness function.

Symbolic regression subsumes linear regression, generalized linear regression, and generalized additive models into a larger class of methods. Such methods have been used with success to infer equations of dynamical systems directly from data [Q5, Q6, Q7, Q8, Q9]. One problem with deterministic chaotic systems is the sampling of phase space using embedding [Q10, Q11]. For a high-dimensional system, this leads to prohibitively long sampling times. Typical reconstruction methods use delay coordinates and the associated differences, this results in mapping models for the observed systems. Mathematically, differential coordinates are better suited for modeling but they are not always accessible from data. Both approaches, difference and differential embedding, are discussed in [Q12] with numerical methods to obtain suitable differential variables from data. Modern methods like diffusion maps [Q13, Q14] or local linear embedding [Q15], including the analysis of stochastic systems, circumvent the curse of dimensionality by working directly on the manifold of the dynamical system.

Symbolic regression has been used in previous works for the prediction and identification of dynamical systems [Q16, Q17, Q18, Q19], more recently it has been used for the control of turbulent flow systems [Q20, Q21]. In this work we use a multiobjective function evaluation mechanism to avoid the above mentioned bloating effect by including minimizing the solution length as an explicit objective in the fitness function. Technically, we combine in our study symbolic regression with a subsequent automatic simplification and multiobjective optimization. As a consequence we can select a complexity and interpret the equations found. We use open-source Python packages for the analysis. For quick tests, we conduct symbolic regression using an elastic net method provided by the Fast Function Extraction (FFX) package [Q22]. A more general, but usually slower method implemented as a GP algorithm based on the `DEAP` package [Q23]. Subsequent simplification is obtained using the `sympy` package [Q24]. All mentioned "packages" are Python software packages, of course, any other programming framework with similar functionality will do as well.

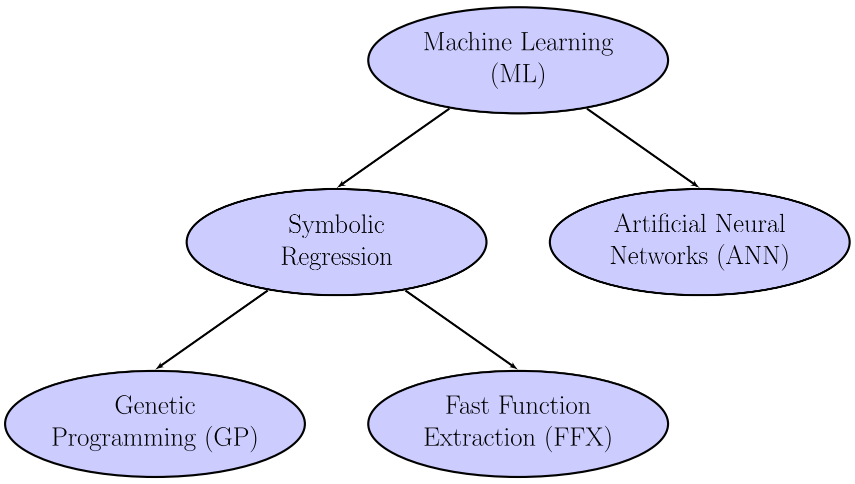


Figure 3.1: Illustration of the relation of the methods mentioned in the introduction. We only sketch a part of machine learning, which is relevant for our work.

For a systematic study we examine numerically-generated data from a harmonic oscillator as the simplest system to be predicted, and a more involved system of coupled FitzHugh-Nagumo oscillators, which are known to produce complex behaviour and may serve as a very simple model for neurons. We investigate the capacity of the ML approach to detect an incoming front of activity, and give exact equations for the regression. We compare different sampling and spatio-temporal embedding methods, and discuss the results: it is shown that a space-time embedding has advantages over time-only and space-only embedding.

Our final example concerns a real-world application, the short-term and medium-term forecasting of solar power production. In principle, this could be achieved trivially by a high-resolution weather forecast and knowledge of the transfer of solar energy to solar cells, a very well-understood process [Q25]. However, such a highly resolved weather forecast does not exist, because it is prohibitively expensive: even the largest meteorological computers are still unable to compute the weather on small spatial scales, let alone with a long time horizon at high accuracy. As the dynamical systems community identified a long time ago, this is mainly due to uncertainties in the initial conditions, as demonstrated by the celebrated Lorenz equations [Q26]. Consequently, we follow a data-based approach and improve upon weather predictions using local energy production data as a time series. We are aware that use of the full set of weather data will improve the reported forecast, but increasing the resolution is not our interest here, rather the proof of concept of the ML method and its applicability to real-world problems.

The rest of this paper is organized as follows. In Section 3.2 we discuss the methods followed by details of our implementation in Section 3.3. This section is followed by a longer Section 3.4 where results are presented for the above-mentioned example systems. We end the paper with a summary and conclusions, Section 3.5.

3.2 METHODS

In the field of dynamical systems, and in particular nonlinear dynamical systems, reconstruction of the characteristics of an observed system from data is a fundamental scientific topic.

In this regard, one can distinguish parameter and structure identification. We first discuss the existing literature on parameter identification which is easier in that there is an established mathematical framework to fit coefficients to known curves representing experimental data, which in turn result from known dynamics. This can be conducted for linear or non-linear functions. For deterministic systems, with the advent of modern computers, quantities like fractal dimensions, Lyapunov exponents and entropies can also be computed to make systems comparable in dynamics [Q27, Q28]. These analyses further allow the rough characterization of the type and number of orbits of a dynamical system [Q29]. On the other hand, embedding techniques have been developed to reconstruct the dynamics of a high-dimensional system from lower-dimensional time series [Q30, Q10, Q11].

These techniques have a number of limitations with respect to accuracy [Q31] and the amount of data needed for making good predictive models. A chaotic system with positive Lyapunov exponents has a prediction horizon which depends heavily on accuracy and precision [Q31] of the data, since chaos "destroys" information. This can be seen very clearly by the shift map example [Q28]. However a system on a regular orbit, even marked with complicated equations, might be predicted accurately. In high-dimensional systems, one needs a large amount of data to overcome the "curse of dimensionality" [Q27]. In fact it can be shown that for each dimension, the number of data needed increases on a power-law basis [Q27, Q32]. Eventually, the direct inference of the underlying equations of motion from data can be approached using regression methods, like Kalman filtering, Generalized Linear Models (GLMs), Generalized Additive Modelss (GAMs), or even more general schemes, see [Q33] and references therein. Apart from the equations themselves, partial derivatives often have to be estimated [Q12], which is an additional problem for low-precision data.

For structure identification, a more complicated task, in the last 10-15 years, powerful new methods from computer science have been applied. This includes numerous studies on diffusion maps, local linear embedding, manifold learning, support vector machines, artificial neural networks, and symbolic regression [Q15, Q13, Q34, Q16]. Here, we focus on symbolic regression. It must be emphasized that most methods are not unique and their success can only be tested based on their predictive power.

3.2.1 *Symbolic Regression*

One drawback of many computational-oriented methods is the lack of equations that can be analyzed mathematically in the neighborhood of analyzed trajectories. Symbolic regression is a way to produce such equations. It includes methods that identify the structure or parameters of the searched equation or both of them simultaneously with respect to objective functions.

This means that methods like GLM, or GAM are contained in such a description. A recent implementation of GLMs is FFX [Q22], which is explained briefly in 3.2.1. Genetic programming, explained in Section 3.2.1, is another intuitive method and often used for symbolic regression. Here, the algorithm searches the function space through random combinations and mutations of functions, chosen from a basic set of equations.

Symbolic regression is supposed to be form free and thus unbiased towards human perception. However, human knowledge enters in the meta-rules imposed on the model through the basic building blocks and rules on how they can be combined. Thus, the optimal model is always conditioned on the underlying meta-rules.

Genetic Programming

Genetic Programming (GP) is an evolutionary algorithm to find an optimal algorithm or program. The term "programming" in optimization is used synonymously with "plan" or algorithm. It was used first by Dantzig, the inventor of linear programming, at a time when computer programs did not exist as we know them today [Q35]. The algorithm seeks an optimal algorithm, in our case a function, using evolutionary, or "genetic" strategies. The pioneering work was established by [Q36]. We can briefly describe it as follows: in GP we can represent formulae as expression trees, such as that shown in Fig. 3.2. Non-terminal nodes are filled with elements from a basic function set defined by the meta-rules. Terminal nodes consist of variables or parameters. Given the optimization problem

$$f^* = \underset{f}{\operatorname{argopt}}\,\Gamma \tag{3.1}$$

we seek the optimal solution f^* through optimizing (minimizing or maximizing, or for some cost functionals, finding the supremum or infimum) the fitness (or cost) functional Γ. To find the optimal solution, GP uses a whole population of candidate solutions in parallel which are evolved iteratively through fitness proportionate selection, recombination and mutation operations. The initial generation is created randomly. Afterwards, the algorithm cycles through the following loop until it reaches its convergence or stopping criteria:

- **breed**: Based on the current generation G_t, a new set of size λ of alternative candidate solutions, the offspring O_t, are selected. Several problem-dependent operators are used for this tweaking step, e. g. changing parts of a candidate solution (mutation) or combining two solutions into two new ones (crossover). These tweaking operations may include selection pressure, so that the "fitter" solutions are more likely to produce offspring.
- **evaluate**: The offspring O_t are evaluated, i.e. their fitness is calculated.
- **select**: Based on the fitness value, members of the next generation are selected.

This scheme fits the requirements of symbolic regression. Mutation is typically conducted by replacing a random subtree by a new tree. Crossover takes two trees and swaps random subtrees between them. This procedure is illustrated in Fig. 3.2. The fitness function uses a typical error metric, e. g. least squares or normalized root mean squared error.

The random mutations sample the vicinity of their parent solution in function space. As a random mutation could likely lead to less optimal solution, it does not ensure a bias towards optimality. However, this is achieved by the selection, because it ensures that favourable mutations are kept in the set while others are not considered in further iterations.

By design and when based on similar meta-rules, GP includes other algorithms like GLM or linear programming [Q34].

Algorithm 1 Top level description of a GP algorithm

```
procedure MAIN
    $G_0 \leftarrow \text{random}(\lambda)$
    $\text{evaluate}(G_0)$
    $t \leftarrow 1$
    repeat
        $O_t \leftarrow \text{breed}(G_{t-1}, \lambda)$
        $\text{evaluate}(O_t)$
        $G_t \leftarrow \text{select}(O_t, G_{t-1}, \mu)$
        $t \leftarrow t + 1$
    until $t > T$ or $G_t = \text{good}()$
end procedure
```

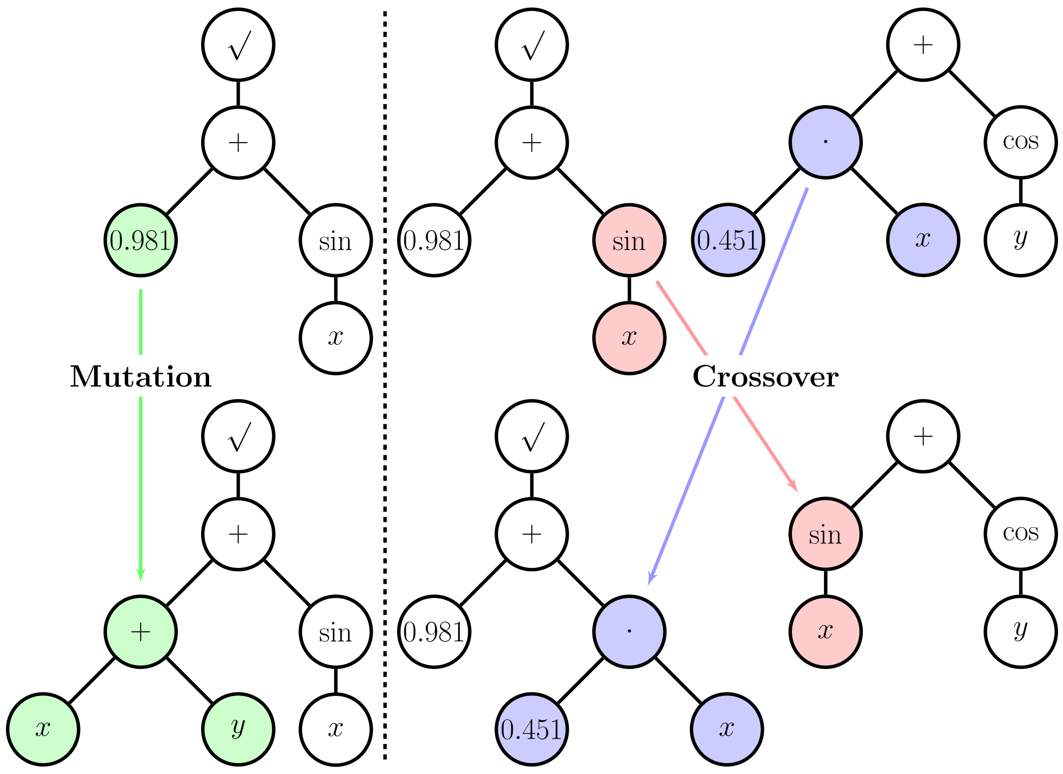


Figure 3.2: Illustration of the genetic programming mutation and crossover. The upper left expression tree describes the function $f(x, y) = \sqrt{0.981 + \sin(x)}$. Mutation is conducted by picking a random subtree, here the single terminal node 0.981 and replacing it with a new random expression tree. Similarly, the crossover operator (right) takes two expression trees and swaps two random subtrees.

FFX and the Elastic Net

Here we briefly summarize the FFX algorithm of McConaghy *et al.*[Q22]. This is a symbolic regression algorithm based on a combined generalized linear model and elastic net approach:

$$f(\vec{x}) = a_0 + \sum_{i=1}^{N_B} a_i \phi_i(\vec{x}) \tag{3.2}$$

where $\{a_i\}$ are a set of coefficients to be determined, and $\{\phi_i\}$ are an overdetermined set of basis functions described by an heuristic, simplicity-driven set of rules (e. g. highest allowed polynomial exponent, products, non-linear functions, ...).

In the elastic method, a least squares criterion is used to solve the fitting problem. To avoid overfitting, i.e. high model sensitivity on training data, two regularizing terms are added: The ℓ_1, and ℓ_2 norms of the coefficient vector. The ℓ_1 norm favors a sparse model (few coefficients) and simultaneously avoids large coefficients. The ℓ_2 norm ensures a more stable convergence as it allows for several, possibly correlated variables instead of a single one. The resulting objective function written in its explicit form reads [Q37]:

$$\vec{a}^* = \underset{\vec{a}}{\operatorname{argmin}} ||y - f(\vec{x}, \vec{a})||_2 + \eta\rho||\vec{a}||_1 + (1-\rho)\lambda||\vec{a}||_2 \tag{3.3}$$

where y are the data, $\eta \geq 0$ the regularization weight and $\rho \in [0, 1]$ is the mixing between ℓ_1 and ℓ_2 norms. A benefit of the regularized objective function is that it implicitly gives rise to models with different complexity, i.e. different number of bases N_B.

For large values of η, the predicted coefficients will all be zero. Reducing λ will result in more complicated combinations of non-zero coefficients. For every point on the (η, ρ)-grid, the "elastic net", one can obtain a single optimal model using a standard solver like coordinate descent to determine the optimal coefficients $\vec{a}^*$.

A small change in the elastic net parameters leads to a small change in $\vec{a}^*$ such that one can use the already obtained solution of a neighboring grid point to restart coordinate descent with the new parameters.

For the obtained models we can calculate the normalized root mean-squared error and model complexity (number of used basis functions). The FFX algorithm is based purely on deterministic calculations. Hence its runtime compared to a similar GP algorithm is significantly shorter. However, the meta-rules are more stringent.

3.2.2 *Multiobjective Fitness*

As mentioned in Section 3.2, the solution of the regression problem is not unique in general. A major factor which motivates symbolic regression is its comprehensible white-box nature opposed to the black-box nature of, for example neural networks. Invoking Ockhams razor ("law of parsimony"), a simple solution is considered superior to a complicated one [Q38, Q39] as it is more easy to comprehend. In addition, more complicated functions are prone to overfitting. This means that complexity should be a criterion in the function search, such that more complex functions are considered less optimal. We therefore seek a solution which satisfies two objectives.

Comparing solutions by more than one metric Γ_i is not straightforward. One possible approach is to weight these metrics into one objective Γ:

$$\Gamma = \sum_i^N w_i \Gamma_i \tag{3.4}$$

making different candidate solutions easily comparable. In this expression, it is implicitly assumed *a priori* that there is a linear trade-off between the individual objectives. An example for such a composite metric is the elastic net, described by Equation 3.3. This approach has three major flaws:

- One needs to determine suitable (problem dependent) w_i.
- One does not account for non-linear trade-offs (e. g. all-or-nothing in one objective).
- Instead of single optimal solution there may be a set of optimal solutions defining the compromise between conflicting objectives (here $\Gamma_1 \simeq$ error *versus* $\Gamma_2 \simeq$ complexity).

The optimal set is also called the Pareto-front, it describes the set of points $(\Gamma_1, \ldots, \Gamma_N)$, where at least one of the objectives Γ_i is minimum. This set is called as well set of non-dominated candidate solutions, i.e. candidate solutions that are not worse than any other solution in the population when compared on all objectives. We illustrate the pareto front Fig. 3.3 by filled circles. For the FFX algorithm, one can obtain the (Pareto-) optimal set of candidate solutions by sorting the models. The mapping from parameter space to the Pareto-optimal set is called Pareto-filtering.

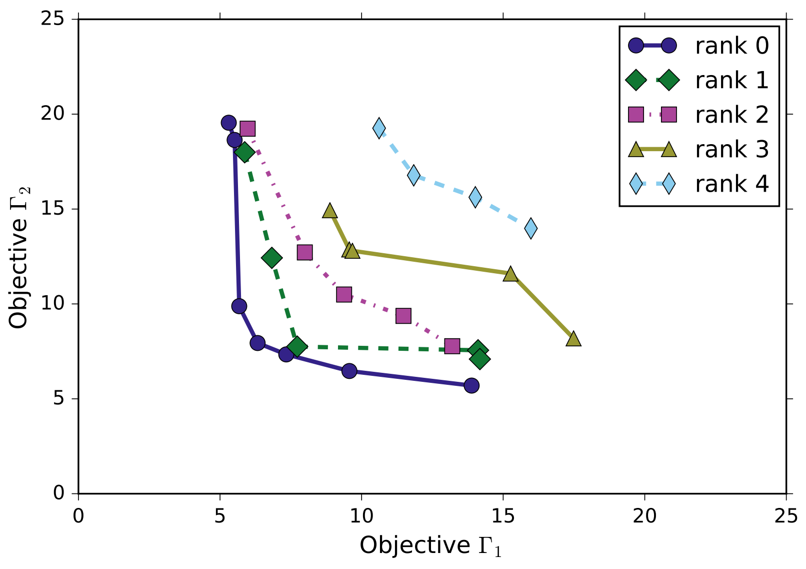


Figure 3.3: Illustration of the ranking for Pareto optimization. For a given set of points in the "objectives plane"' (Γ_1, Γ_2) one labels first all points with rank zero that possess at least one minimum objective, the Pareto front (filled circles, solid line). Then, these points are eliminated from the set and the remaining points are ranked the same way: the rank 1 model emerges (filled diamonds, dashed line). This procedure is repeated until all points are ranked

Interestingly, the concept of non-domination already partly solves the sorting problem in higher dimensions as it maps from $\mathcal{R}^N$ to M ordered one-dimensional manifolds. A sorting algorithm must use a kind of ranking, which in this case is described as follows: Candidate solutions in the Pareto-front are of rank 0. Solutions of rank 1 are obtained as follows: of all models found one subtracts the rank 0 ones, and determines a new Pareto front for the remaining models. The models on this front have rank 1. This procedure can be continued until all candidate solutions are ranked. Formally, we introduce the generalized comparison operator $\succ$, and define Model 1 f_1 to be better than Model 2 f_2 if its rank is lower:

$$f_1 \succ f_2 \Longleftarrow \text{rank}(f_1) < \text{rank}(f_2)$$

To compare models of the same rank, one has to introduce an additional heuristic criterion, for which there are several choices [Q40, Q41, Q42]. Usually the criterion promotes uniqueness of a candidate solution to ensure diversity of the population to avoid becoming trapped in a local minimum. As the uniqueness of a solution may depend on its representation and is usually costly to compute, often its projection to fitness space is used. This is conducted to ensure an effective spread of candidate solutions on the Pareto-front.

For example, the Non-Dominated Sorting Algorithm II (NSGAII) [Q40] uses a heuristic metric called crowding distance to compare two models of the same rank. The scaled Euclidean distance in fitness space to the neighboring models is used to describe the uniqueness of a model. For NSGAII we have:

$$f_1 \succ f_2 \Longleftarrow \begin{cases} \operatorname{rank}(f_1) < \operatorname{rank}(f_2) \\ \operatorname{rank}(f_1) = \operatorname{rank}(f_2) \text{ and} \\ \quad \operatorname{distance}(f_1) > \operatorname{distance}(f_2) \end{cases} \tag{3.5}$$

Out of the current generation and their offspring $G_t \cap O_t$ the μ best, in terms of $\succ$, solutions are chosen for the next generation G_{t+1}. This selection method ensures elitism, i.e. the best solutions found so far are carried forward in next generations. Looking at the high-level description in 1, G_t can be seen as an archive which keeps old members as long as they are not dominated by a new solution from the current offspring O_t.

The different selection strategies were first studied in the context of genetic algorithms, but more recently they have been successfully applied to symbolic regression [Q43, Q44].

3.3 OUR GP SETUP

For all applications presented in Section 3.4, our function set is $\{+, *, -, /, \sin, \cos, \exp, \log, \sqrt{}, {}^2\}$. All discontinuities are defined as zero. Our terminal set consists of the input data x_i as well as symbolic constants c_i which are determined during evaluation. We set up our multiple objectives as follows: the algorithm runs until the error of the most accurate model is below 0.1%, or for 100 generations. The population size μ as well as the number of offspring per generation λ is set to 500. The depth of individuals of the initial populations varies randomly between 1 and 4. With equal probability we generate the corresponding expression trees where each leaf might have a different depth or each leaf is forced to have the same depth. For mutation we randomly pick a subtree and replace it with a new tree, again using the half and half method, with minimum size 0 and maximum size 2. Crossover is conducted by randomly picking a subtree each and exchanging them. Our breeding step is composed of randomly choosing two individuals from the current population, performing crossover on them with probability $p = 0.5$ and afterwards always mutating them. Our multiobjective cost functional has the following components

$$\Gamma_1 = \text{NRMSE}\,(y, \hat{y}) = \frac{\sqrt{\sum\limits_{i=1}^{N} \frac{(y_i - \hat{y}_i)^2}{N}}}{y_{max} - y_{min}} \tag{3.6}$$

where Normalized Root Mean Squared Error (NRMSE) is the normalized root mean-squared error of the observed data y and its predictor $\hat{y} = f(\vec{x})$, and

$$\Gamma_2 = \operatorname{size}(f) \tag{3.7}$$

is simply the total number of nodes in the expression tree f. Selection is conducted according to NSGAII. In this paper, a model is called accurate if its error metric Γ_1 is small, where "small" depends on the context. For example, numerical data might be modeled accurately if $\Gamma_1 \leq 0.05$ and measured data might be modeled accurately if $\Gamma_1 \leq 0.20$. Similarly a model is complicated if its complexity Γ_2 is relatively large. "Good" and its comparatives are to be understood in the sense of $\succ$.

During the generation of the initial population and selection, we force diversity by prohibiting identical solutions. It is very unlikely to randomly create identical solutions. However, offspring may be nearly identical in structure as well as fitness and consequently a crossover between parent and child solution may produce an identical grandchild solution. The probability of such an event grows exponentially with the number of identical solutions in a population and therefore it lowers the diversity of the population in the long-term risking a pre-mature convergence of the algorithm. Thus, by prohibiting identical solutions, the population will have a transient period until it reaches its maximum capacity. This will also reduce the effective number of offspring per generation. This change reduces the probability of becoming trapped in a local minimum because of a steady state in the evolutionary loop.

Our main emphasis is the treatment of the model parameters c_i. In standard implementations, e. g. the already mentioned [Q43, Q44], the parameters are mutated randomly, like all other nodes. Here, using modern computational power we are able to use traditional parameter optimization algorithms. Thus, the calculation of Γ_1 becomes another optimization task given the current model f_j:

$$\Gamma_1 = \text{NRMSE}\left(y, f(\vec{x}, \vec{c}^*)\right) \tag{3.8}$$

with

$$\vec{c}^* = \underset{\vec{c}}{\text{argmin}}\, \text{NRMSE}\left(y, f(\vec{x}, \vec{c})\right) \tag{3.9}$$

The initial guess for c_i is either inherited or set to one. Thus, we effectively have two combined optimization layers. Each run is conducted using 10 restarts of the algorithm. The Pareto front is the joined front of the individual runs. Finally, we can use algebraic frameworks to simplify the obtained formulae. This is useful, since a formula (phenotype, macrostate) may be represented by many different expression trees (genotypes, microstates).

3.4 CASE STUDIES

We present here results for three systems with increasing difficulty: first, we demonstrate the principles using a very simple system, the harmonic oscillator (A); second, we infer a predictive model for a set of coupled oscillators (B); and finally we show how we can predict a very applied system, namely the power production from a solar panel (C). For the first two examples we use numerically produced data, where we have full control over the system, while for the demonstration of applicability we use data from a small solar power station [Q45].

3.4.1 *Harmonic Oscillator*

In this subsection we describe the first test of our methodology: an oscillator should be identified correctly and a accurate prediction must be possible. Consequently, we investigate the identification of a prediction model, not necessarily using a differential formalism. This might be interpreted as finding an approximation to the solution of the underlying equation by data analysis. A deep investigation of the validity of the solution for certain classes of systems is rather mathematical and is beyond the scope of this investigation.

Our system reads

$$\dot{x} = y \tag{3.10}$$

$$\dot{y} = -\omega^2 x \tag{3.11}$$

where x and y are the state variables and ω is a constant. We use the particular analytical solution $x(t) = x_0 \sin(\omega t)$, $y(t) = x_0\omega\cos(\omega t)$. The prediction target is $x(t+\tau)$, where τ is a time increment.

Since the analytical solution is a linear combination of the feature inputs, just $N = 2$ data points are needed to train the model. This holds for infinite accuracy of the data and serves as a trivial test for the method. In general, a learning algorithm is "trained" on some data and the validity of the result is tested on another set, that is as independent as possible. That way, overfitting is avoided. For the same reason one needs to define a stop criterion for the algorithm, e. g. the data accuracy is 10^{-5}, it is useless and even counterproductive to run an algorithm until a root mean square error of 10^{-10} (the cost function used here) is achieved. For the example under consideration, we stop the training once the training error is smaller than 1.

Typically, a realistic scenario should include the effect of noise, e. g. in the form of measurement uncertainties. We consequently add "measurement" Gaussian noise with mean zero and variance proportional to the signal amplitude: $\xi_1 \sim \mathcal{N}(0, (\sigma x_0)^2)$, $\xi_2 \sim \mathcal{N}(0, (\sigma x_0 \omega)^2)$, hence $\tilde{x} = x + \xi_1, \tilde{y} = y + \xi_2$. The training and testing data sets were created as follows: the data are generated between $[0, t_{max}]$. Out of the first half, we chose N values at random for training. For testing purposes we use the second half. We study the parameter space (N, τ, σ) and average the testing errors over 50 realizations for each parameter set. We only present the results obtained with the FFX method. Of course, a similar study using GP is possible, but practically limited by the runtime. However, the generic nature GP allows for stochastic modeling, which is subject of ongoing research. Nevertheless, a preview with qualitatively similar results can be found in Fig. 3.16. In Fig. 3.4 we display the normalized root mean squared error of the prediction using FFX (measured against the noisy data) as a function of the noise amplitude. Given $x(t)$ and $y(t)$ the analytical solution for the non-noisy system is just a linear combination, i.e. $x(t+\tau) = \cos(\omega\tau)x(t) + \frac{\sin(\omega\tau)}{\omega}y(t)$, and has a complexity of two. During training we aim for a NRMSE of 1%. Thus, we find the analytical solution in the limit of small noise amplitude σ, see Fig. 3.4 and Fig. 3.6. Strong noise covers the signal and thus the error saturates.

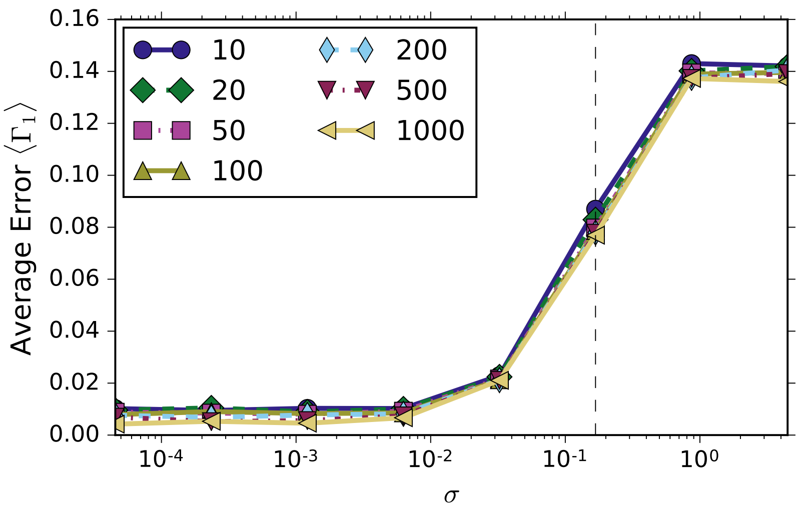


Figure 3.4: Harmonic oscillator study, method FFX: NRMSE (3.6) versus noise level σ for different training set lengths N and fixed $\tau = 10$. Sufficiently small noise does not worsen the predictability, i.e. the prediction algorithm stops at the target training NRMSE of 1%. After 0.3 the error does not increase further, since the noise covers the signal completely. Dashed line marks $\sigma = 0.17$.

The length of the analyzed data is another important parameter: typically one expects convergence of the error $\sim \frac{1}{\sqrt{N}}$ for more data. A "vertical" cut through the data in Fig. 3.4 is shown in Fig. 3.5. The training set length N has a much lower impact than the classical scaling suggests. Crucial for this scaling is the form free structure as well as the heuristic which is used to select the final model. For demonstration purposes, we chose the most accurate model on the testing set, which is of course vulnerable to overfitting. The average complexity, calculated by Equation 3.7 of the final model as a function of the noise amplitude, is shown in Fig. 3.6. As evident we can recover the three regimes of Fig. 3.4. For small noise, the analytical and numerical solution agree. In the intermediate regime we find on average more complex models (in comparison to the analytical solution). Very strong noise hides the signal and a good prediction is impossible. The optimal solution tends to be single constant, i.e. for high σ the complexity tends to smaller values as seen in Fig. 3.6. The prediction error has two components: 1) given a structure, noisy data will lead to uncertain parameters and 2) due to the form-free nature of symbolic regression, noisy data will also lead to an uncertain structure, increasing the uncertainty in the parameters. Thus, final model selection has to be performed carefully, especially when dealing with noisy data. A detailed study is presented for the example of coupled oscillators.

3.4.2 *Coupled Oscillators*

The harmonic oscillator is an easy case to treat with our methods. Now, we extend the analysis to add a spatial dimension. We study a model of FitzHugh-

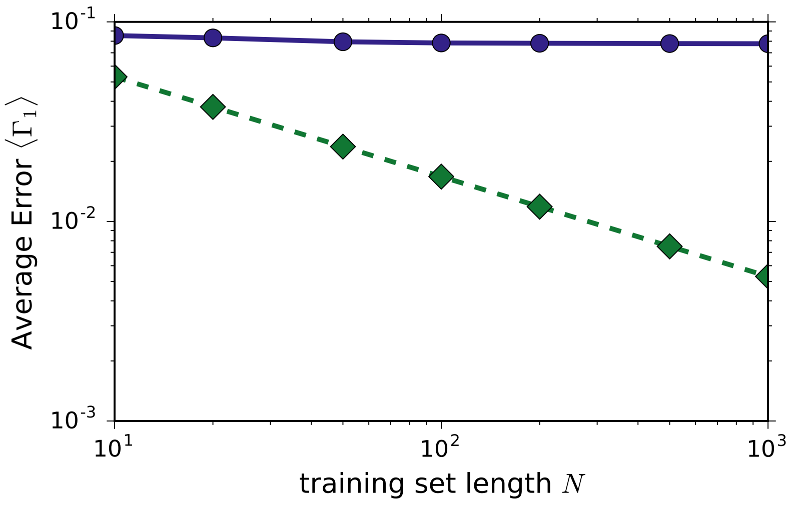


Figure 3.5: Harmonic oscillator study, method FFX: In solid blue: normalized root mean squared error vs training set length N for $\sigma = 0.17$. Dashed green: $e^{-2}/\sqrt{N}$. The error decreases slightly with N, but the scaling is much less rapid than $1/\sqrt{N}$.

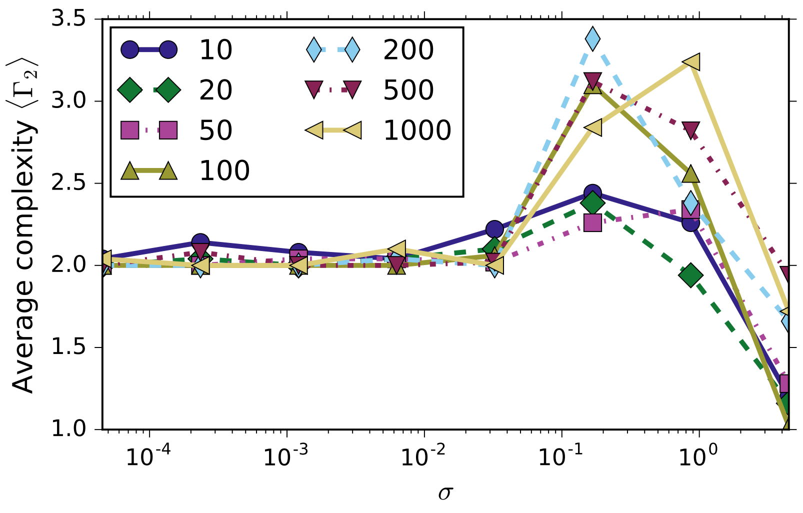


Figure 3.6: Harmonic oscillator study, method FFX: Average complexity of the chosen model vs. noise amplitude σ. The form-free structure allows for overfitting. For small noise, the true solution with complexity 2 is found, for higher noise levels, the algorithm starts to fit the noise and more terms are added, reflected by a higher complexity.

Nagumo oscillators [Q46] on a ring. The oscillators are coupled and generate traveling pulse solutions. The model was originally derived as a simplification of the Hodgkin-Huxley model to describe spikes in axons [Q47], and serves nowadays as a paradigm for excitable dynamics. Here, its spiky behavior is used as an abstraction of a front, observed in real world applications like the human brain, modeled by connected neurons, or a wind power plant network where fronts of different pressure pass through the locations of the wind power plants. The aim is to show that temporal and/or spatial information on the state of some network sites enables an increase in predictability of a chosen site or eventually (if there are waves in the network) to the front detection. The model for the *i*th oscillator is:

$$\dot{v}_i = v_i - \frac{v_i^3}{3} - w_i + I_i + D\sum_{i,j} A_{ij}(v_j - v_i) \tag{3.12}$$

$$\dot{w}_i = \varepsilon(v_i + a - bw_i) \tag{3.13}$$

where v_i and w_i, $i, j = 1, \ldots, N$, denote the fast and slower state variables, I_i is an external driving force, D is the coupling strength parameter, and $A_{ij} \in \{0, 1\}$ describes the coupling structure between nodes i and j. The constant parameters ε, a and b determine the dynamics of the system as ε^{-1} is the time scale of the slower "recovery variable", and a, and b set the position of the fixed point(s). For A_{ij} we choose diffusive coupling on a ring, i.e. periodic boundary conditions. With the external current I_i we can locally pump energy into the system to create two pulses which will travel with the same speed but in opposite directions, annihilating when they meet.

Using different spatio-temporal sampling strategies, the aim is to detect and predict the arrival of a spike train at a location far enough away from the excitation center (i.e. farther than the wave train diameter). We mark this special location with the index zero.

Note that we do not aim to find a model for a spatio-temporal differential equation, since this would involve the estimation of spatial derivatives, which in turn require a fine sampling. This is definitely not the scope here. Rather we focus on the more application-relevant question to make a prediction based on an equation.

The construction of the data set was similar to the single oscillator case: sensors were restricted to the v_i variables. We can record the time series of v_0 and use time delayed features for the prediction. Another option is to use information from non-local sensors.

We prepare and integrate the system as follows: we consider a ring of $N = 200$ oscillators. The constants are chosen as $a = 0.7$, $b = 0.8$, $\tau = 12.5$ and $D = 1$. The system is initialized with $v_i(0) = 0$ and $w_i(0) = -1.5$. With the characteristic function $\chi_T(x) = 1$ if $x \in T$ else 0 we can write the space and time dependent perturbation as $I_i(t) = 5\chi_{t-\lfloor t\rfloor \leq 0.4}(t)\chi_{t\leq 40}(t)\chi_{i\in\{-50,-49\}}(i)$. This periodic perturbation leads to a pair of traveling waves. The data were sampled at times $t_n = n\Delta t$ with $\Delta t = 0.1$. The system has multiple time scales: two are associated with the on-site FitzHugh-Nagumo oscillator ($\tau_{fast} = 1$, $\tau_{slow} = \frac{1}{\varepsilon}$), while two more are due to diffusive coupling ($\tau_{Diff} = D$) and perturbation (τ_{Pert} behaves as $I_i(t)$). The temporal width of the pulse traveling through a particular site, $\tau_P = 8.4$, corresponds to the full width half maximum of the pulse. In Fig. 3.7 we show the evolution of the oscillator network. The state of v_i is color-coded. The horizontal width of the yellow stripe corresponds to the spatial pulse width $\xi \simeq 10.75$. The speed of the spike or front is consequently $v_{front} \sim \xi/\tau_P = 1.28$. An animation of this can be found in the supplemental material. The training data were recorded in three different ways:

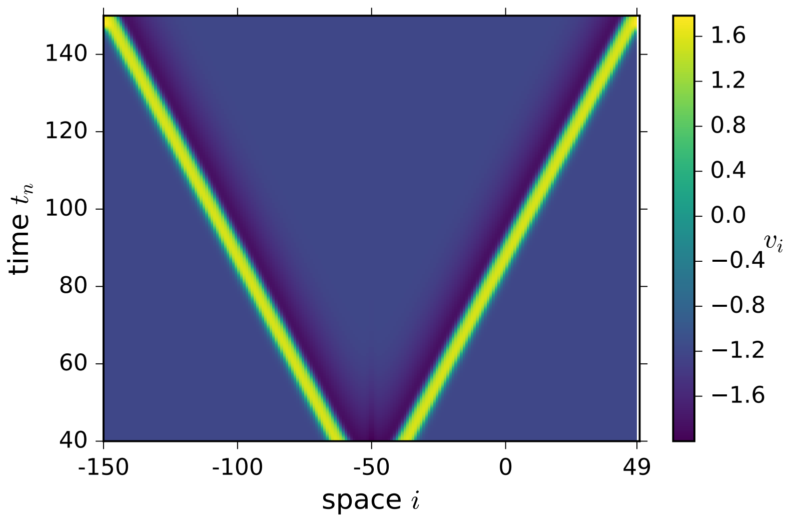


Figure 3.7: Space-time plot of the pulse evolution. v_i is color coded. The front velocity is $v_f = 1.28$. Pulse width (full width half maximum) $\tau_P = 8.4$

- **site-only**: Only v_0 is recorded, and time-delayed features $v_{0,\Delta n} = v_0(t = (n - \Delta n)\Delta t)$ are also included with $\Delta n \Delta t = -1, -2, -3, -4$.
- **spatially extended**: We record v_0 and additionally v_i with $i = -2, -4, \ldots, -10, -20$ (upstream direction).
- **mixed**: This combines the two approaches above. For each site we also include the time delayed features.

To avoid introducing additional symbols we use state variables with double subscripts for discrete times, where the second index refers to time, and one subscript for continuous time. The respective useage is evident from the context. We choose to predict the state at time $t = 2$ given the data described above. In other words, the prediction target is $v_0(t_n + \tau)$ with $\tau = 20 \simeq 2.5\tau_P$, corresponding to the requirement to be far engouh from the excitation point. Of course, this implies a distance of $\Delta x \sim 2.5\xi$. The testing and training sets were selected by using every second point of the recorded time series.

FFX Results

We first discuss the results obtained by FFX (3.2.1). In Fig. 3.8 we display the Pareto fronts using the three different approaches for the training set. All curves have one point in common which represents the best fitting constant (complexity 0). As one would expect, the site only data do not contain enough information to detect a front. Thus, even high complexity models cannot reach an error below 4% and the required error of 1% is never met. In the two other datasets the algorithm has the possibility to find a combination of spatial amd temporal inputs to account for the front velocity. Note that the shape of the front strongly depends on the internal ρ parameter of the elastic net Equation 3.3. More information should not lead to a decrease in predictability. Thus, the Pareto front of a data set richer in features dominate the corresponding Pareto front of a data set with less features. Counter-intuitively, using $\rho = 0.95$ [1] the front for the mixed dataset becomes non-convex as some good fitting models are hidden by the regularizer. Thus, we can use ρ to influence the shape of the front. Despite that, the most accurate model of the mixed data set is still the most accurate model overall.

In the following we discuss the results for the best models for each feature set.

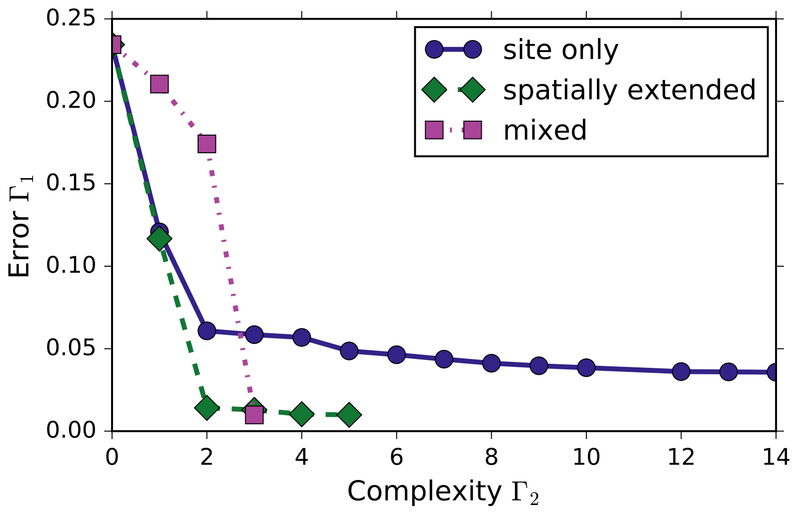


Figure 3.8: Coupled spiking oscillators, method FFX: Pareto fronts for the different spatio-temporal samplings of the network data. For this plot we use $\rho = 0.95$. This leads to the non-convex shape of the front based on the most information. The models are re-evaluated on the testing set.

If we take the perspective of an observer sitting at $i = 0$, we see the spike passing: first the state is zero, then a slow increase is observed followed by a rapid increase and decrease around the spike maximum. Eventually the state returns slowly to zero. Statistically, the algorithm is trained by long quiet times and a short, complicated spike form which is hard to model by a reduced set of state variables. This is illustrated in Fig. 3.9a where for any feature set the biggest differences occur in the spike region. Apparently, the model with site-only variables shows worse results than the spatial one, and the spatio-temporal set models best the passing spike. We note that in a direct confrontation, the true and modeled signal would hard to be distinguished. In Fig. 3.9b we confront the time derivative for the model from mixed variables. The true and modeled spike are indistinguishable by eye.

The formulae of the most accurate models are shown in Table 3.1. For site-only features, quadratic combinations of points at different times occur. This reflects the approximation of the incoming front by a quadratic term. If, however only spatial points are used, the dynamics far away are used to predict the incoming front. If the small terms are neglected, the model consists of the signal at the target site itself, and the previous site (-2) which carries the largest weight. Physically, it means that despite being far away the front is already felt at 2 sites away. Since the front is stationary in a co-moving frame, spatio-temporal embedding

1 The default value of the package which works well in most scenarios.

temporal site-only	$-0.0273 + 3.34v_{0,0} - 2.41v_{0,0}v_{0,-10} - 2.09v_{0,-40}v_{0,-10} + 1.64v_{0,-20}^2 - 1.53v_{0,-20} - 1.16v_{0,-10} + 0.991v_{0,-30}^2 + 0.684v_{0,0}^2 + 0.463v_{0,-30} + 0.433v_{0,-20}v_{0,0} + 0.373v_{0,-20}v_{0,-10} - 0.359v_{0,-40}^2 + 0.216v_{0,-40} + 0.00286v_{0,-10}^2$
spatially extended	$-0.00247 + 0.897v_{-2,0} + 0.178v_{0,0} - 0.0650v_{-4,0} + 0.00280v_{-10,0} - 0.00210v_{-8,0}$
temporal spatial	$0.00894 + 0.442v_{-4,-30} + 0.346v_{-2,-10} + 0.175v_{-2,0}$

Table 3.1: Coupled spiking oscillators, method FFX. Formulae of the most accurate models. The spatio-temporal embedding reproduces the data very well, i.e. an early detection is possible.

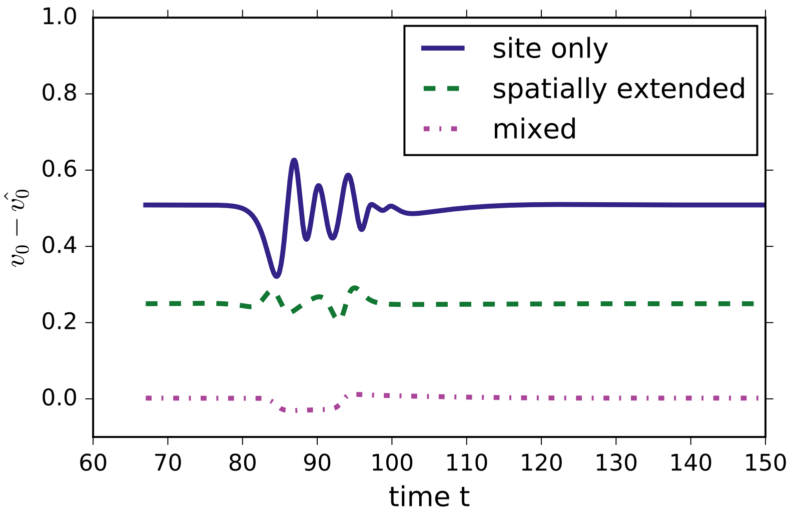
is best, namely sampling the spike train in space and moving in time with the train velocity. Then we have a simple and compact linear dependence as seen in the last row of Table 3.1. Let us inspect the possible physics in the model approximating the constants a_0, a_1, a_2, a_3 roughly as 0, 0.45, 0.35, 0.175 such that $a_2 = 2a_3$. We first notice that $\tau_p = 8.4 \simeq 10$. The last terms can then be recombined to $a_3 v_{-2,-10} + a_3 v_{-2,-10} + v_{-2,0}$ as a mean value of the state with time distance of approximately one typical time scale. The state at -30 is at the backside of the front and together the most important information, namely the increase and decrease of the incoming signal is selected by the model. Alternatively, since $v(0,t) = v(-v_f \tau_P, t - \tau_P)$ the best model in Table 3.1 can be interpreted as the weighted average of the closest combination $(\Delta i, \Delta t)$ to represent the front velocity $(\frac{\Delta i}{\Delta t} = \frac{4}{3} \approx v_f)$. This demonstrates how powerful the algorithm works in selecting important features.

GP Results

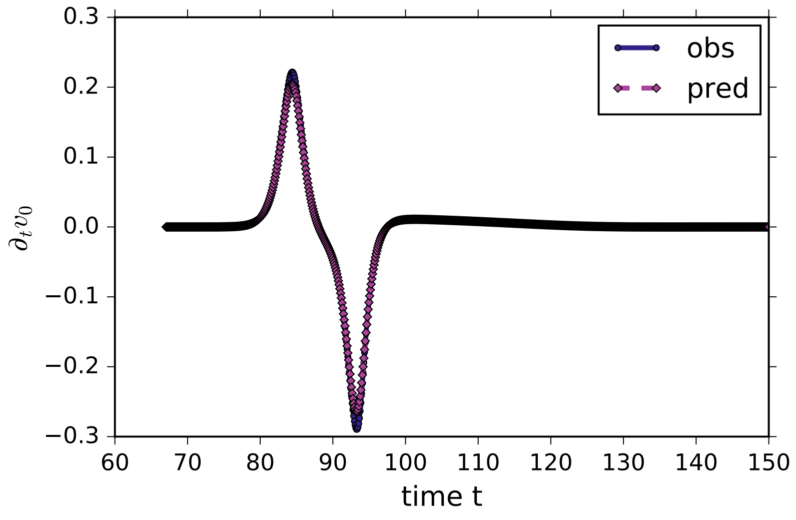
We again examine the Pareto-optimal models illustrated in Fig. 3.10. For each feature set we obtain a non-convex Pareto front. The shape and the values of the fronts are broadly similar to the results obtained by FFX. Because GP is an evolutionary method and relies on random breeding rules, we display averaged results: we initialize the algorithm with different seeds of the random number generator, calculate the Pareto fronts and average the errors for the non dominated models of the same complexity. Note that not all complexities occur on each particular front. This way, we obtain a generic Pareto front and avoid atypical models which may occur by chance. The results of Table 3.2 is not averaged, but the best result for one specific seed (42). The errors of the models reachable by the different sets are again decreasing from site only over spatially extended to mixed. However, the mixed model reaches almost zero error which is quite remarkable!

The difference plots for the method are given in Fig. 3.11. While the site only set is not able to give a convincing model for an incoming front, the spatially extended set gives a reasonable model with little error. The mixed model is very good with perfect coincidence of model and true dynamics. This model cannot be distinguished by eye from the observed signal.

The models provided by the GP algorithm with seed 42 are given in Table 3.2. Due to the very general character of GP these can be overwhelming at first glance.



(a)



(b)

Figure 3.9: Coupled spiking oscillators, method FFX. For each feature set, the most accurate model is used as the predictor $\hat{v}_0$. In (a) we show the difference $\delta v_0 = v_0 - \hat{v}_0$. The upper two curves are shifted by 0.25 and 0.5 respectively. In (b) we compare the time derivative (approximated by the finite difference quotient) of the most accurate model overall (mixed) and the real data. For details see text.

However, we can simplify them down by using computer algebra systems like `sympy` [Q24] or `Mathematica` (here we use `sympy`).

The interpretation of the GP results requires a bit more thinking. In essence, they follow a logic similar to the FFX results. The site-only model is complicated, and instead of a square operator a trigonometric function is used to mimic the incoming pulse. Since the data do not include directly all information needed, the algorithm tries to fit unphysical functions. This is clearly a non-deterministic and overfitting result, mirrored by the high complexity of the functions involved. For spatially extended models, we obtain linear and sinusoidal components, and the

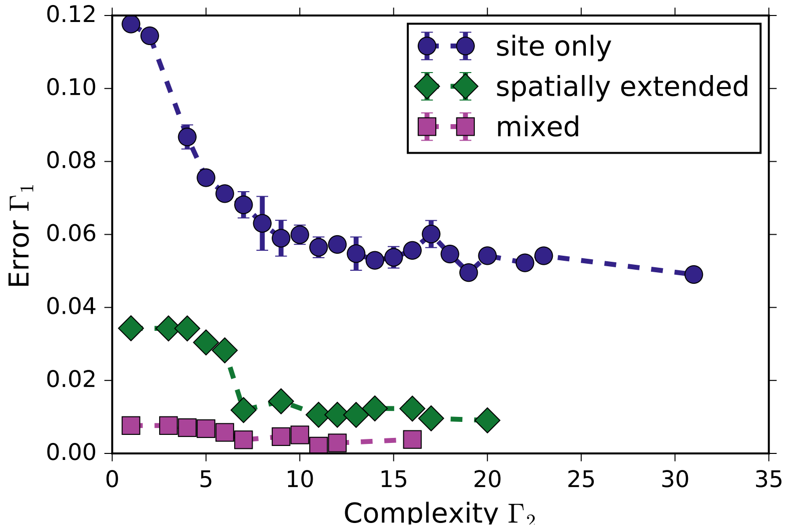


Figure 3.10: Coupled spiking oscillators, method GP. Averaged Pareto fronts, for each spatio-temporal sampling option, 10 runs are conducted and the resulting complexities and errors are averaged. Errorbars represent the standard deviation. For the spatially extended and mixed data sets the errors are smaller than the circle size. The models are re-evaluated on the testing set.

temporal site-only	$v_{0,0}^2/(v_{0,-10}+\sqrt{(}-v_{0,-10}(v_{0,0}-v_{0,-30})(v_{0,-30}/\sin(v_{0,-10}+v_{0,-20})+\exp(v_{0,-30})-\sqrt{(}\sin(v_{0,-30}))+\cos(\sqrt{(}v_{0,-30})v_{0,-40}))))$
spatially extended	$0.208v_{0,0}+0.792v_{-2,0}+0.0274\exp(-v_{-4,0})\sin(v_{-2,0})$
temporal spatial	$0.878v_{-4,-30}+0.124496v_{-4,-40}$

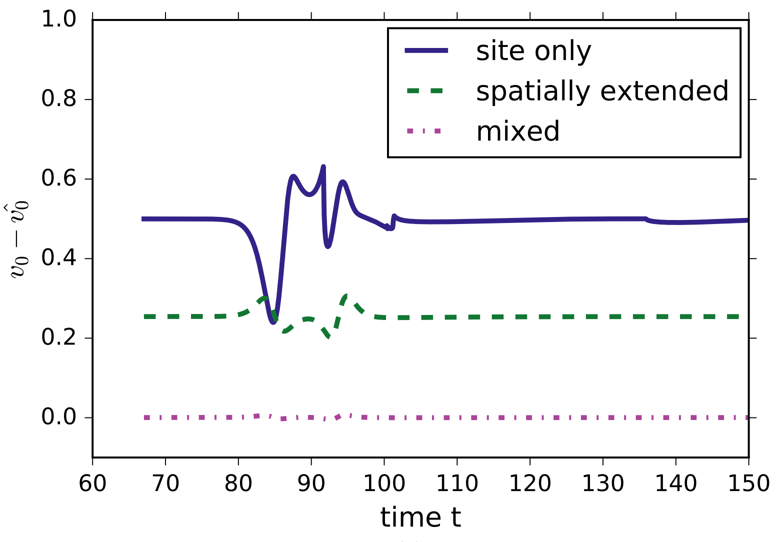
Table 3.2: Coupled spiking oscillators, method GP. Formulas of the most accurate models for seed 42.

model uses only three features, namely the on-site values and the ones at two and four units left on our site under consideration. Remarkably, a sinusoidal behavior is observed with an exponential decrease, which is our intuition. Eventually, the spatio-temporal embedding yields a very simple model which approximates the front velocity v_f to be between 1 and $\frac{4}{3}$. The accuracy of this model is very high.

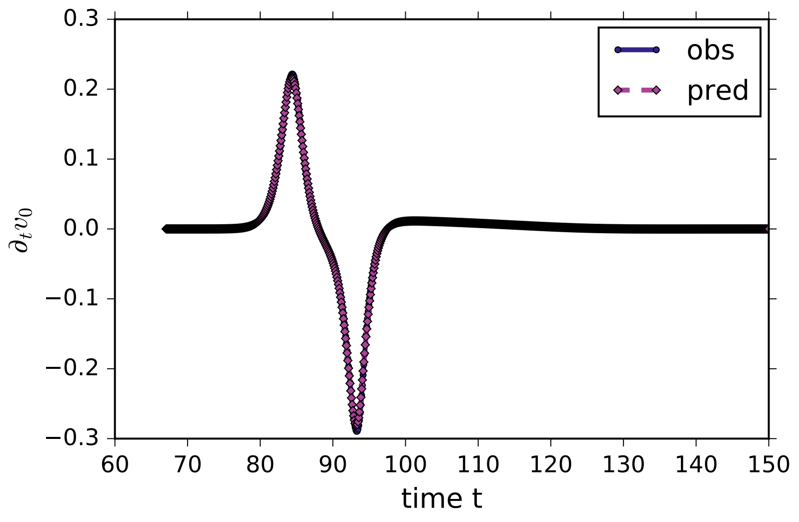
Summarizing, when given enough input information, both methods find a linear model for the predictor $\hat{v}_0(t+\tau)$ by finding the most suitable combination of temporal and spatial shift to mimic the constant front velocity. If this information is not available in the input data, nonlinear functions are used.

3.4.3 *Solar Power Data*

In this section, we describe the results obtained for one-day-ahead forecasting of solar power production. The input data used for training are taken from the unisolar solar panel installation at Potsdam University with about 30 kW installed. Details are found at [Q45]. We join the solar power data with meteorological forecast data from the freely available European Centre for Medium-Range Weather



(a)



(b)

Figure 3.11: Coupled spiking oscillators, method GP. For each feature set, the most accurate model is used as the predictor $\hat{v}_0$. In (a) we show the difference $\delta v_0 = v_0 - \hat{v}_0$. The upper two curves are shifted by 0.25 and 0.5 respectively. In (b) we compare the time derivative (approximated by the finite difference quotient) of the most accurate model overall (mixed) and the real data. Prediction and true data cannot be distinguished by eye.

Forecasts (ECMWF) portal [Q48] as well as the actual observed weather data. These public data are of limited quality and serve for our proof of concept with real data and all their deficiencies.

The solar panel data $P(t)$ were recorded every five minutes, at geoposition 52.41 latitude, 12.98 longitude. The information about the weather can be split into two categories: weather observations of a station near the power source $W(t)$ and the weather forecast $\hat{W}(t+\tau)$, where τ is the time difference to the prediction target. We do not have weather data from the station directly, but can use data from a weather station nearby (ID: 10379). The weather forecast data are obtained every six hours at the closest location publicly accessible, 52.5 latitude and 13 longitude. Typical meteorological data contain, but are not limited to, the wind speed and direction, pressure at different levels, the irradiation, cloud coverage, temperature and humidity. However, in this example, we only use temperature and cloudiness as well as their forecasts as features for our model. The latter is obtained by minimizing

$$\begin{aligned} \Gamma_1 = &\quad \text{NRMSE}\Big(P(t+\tau), \hat{P}\big(P(t), W(t), \hat{W}(t+\tau)\big)\Big) \\ \Gamma_2 = &\quad \text{size}(f) \end{aligned} \tag{3.14}$$

with f the model under consideration. Our prediction target is $\hat{P}(t+\tau)$ with $\tau = 24$, the one-day-ahead power production. We create our datasets with a sampling of 1h. While additional information from the solar power data remains unused, the prediction variables have to be interpolated. The quality of the forecast depends on quality of the weather measurement and weather forecast. As we use publicly available data, we can only demonstrate the procedure and cannot attain errors as low as those used in commercial products, which will be discussed elsewhere. The features of the the data set are listed in Table 3.3. Furthermore, we

Name	Symbol	Source	Sampling	Variable
Solar power	$P(t)$	direct access	10 min	x_1
Total cloud coverage	$\text{tcc}(t)$	Synop	1h	x_4
2 meter temperature	$T(t)$	Synop	1h	x_3
Total cloud coverage prediction	$\text{tcc}_{pred}(t,\tau)$	ECMWF-TIGGE	6h	x_2
2 meter temperature prediction	$T_{pred}(t,\tau)$	ECMWF-TIGGE	6h	x_0

Table 3.3: Solar power study: description of the data set. We use a set of 5 features drawn from different sources with different sampling.

scale each feature to have its minimum equal zero and maximum equal to one. The models are trained with data from June and July of 2014. Testing is conducted for August 2014. To obtain a first impression (assuming no prior knowledge), we calculate the mutual correlation of the data. The power produced the next day is heavily correlated with the predicted solar irradiation. This is a confirmation that the physics involved is mirrored in the model and that weather prediction is good on average. Quantitative statements on the quality of weather prediction is not easy and can be found in the literature [Q48].

GP Results

Let us consider the results of our forecasting with GP shown in Fig. 3.12. The Pareto fronts are shown for both the training and testing set. As presented for the coupled oscillators in Fig. 3.4.2, we have conducted 10 runs with different seeds and display the averaged result. Of course, for the training set (filled diamonds), increasing complexity means decreasing error. We see a strong deviation for very complicated models of the testing data (filled circles). This may be an indication of a small testing sample, or indicate overfitting. The outlier at $\Gamma = 18$ is a result of the particular realization of the evolutionary optimization. With a different setting, e. g. more iterations, or multiple runs such outliers are eliminated. To clar-

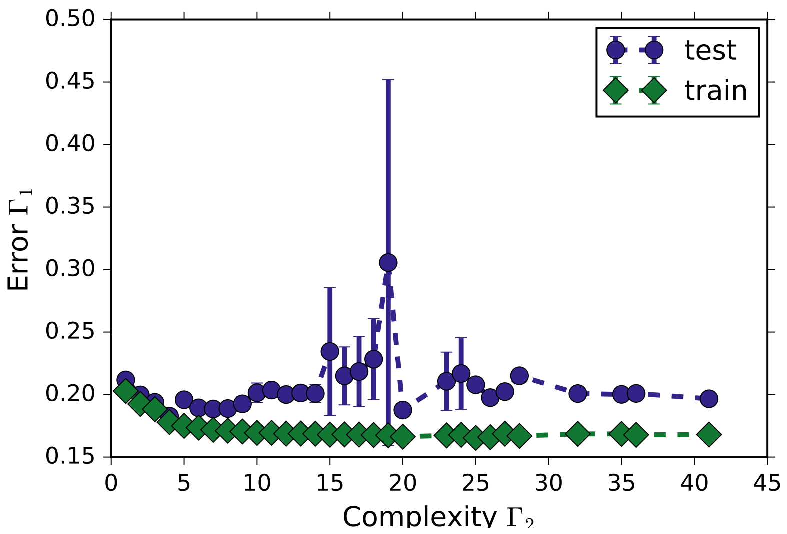


Figure 3.12: Solar power study, Average Pareto front obtained using GP: with increasing complexity, training and testing data first behave similarly, then testing deviates strongly indicating overfitting or too small testing data set, respectively. The peaks around complexity 20 are due to two reasons: there are only few models on the fronts (1-3), and one of them is an extreme outlier.

ify this question, we show the functions found as solution of our procedure with increasing complexity and one specific seed (42) in Table 3.4.

Complexity Γ_2	Error Γ_1	Formula
1.0	0.2117	x_1
2.0	0.1997	$\sin(x_1)$
3.0	0.1938	$\sin(\sin(x_1))$
4.0	0.1827	$0.662\sqrt(x_1)$
5.0	0.1993	$\sqrt(x_0 \sin(x_1))$
6.0	0.1931	$\sqrt(\sin(x_0)\sin(x_1))$
7.0	0.189	$\sqrt(x_0 x_1 \cos(x_3))$
8.0	0.1943	$\sqrt(x_1)(x_0 - x_3 + 0.649)$
9.0	0.2348	$\sqrt(x_1)\cos(x_2 x_3 / x_0)$
10.0	0.192	$\sqrt(x_1)(x_0 - x_4(x_3 - 0.699))$
12.0	0.2057	$\sqrt(x_1)(x_0 - x_2(x_2 x_3 - 0.592))$
16.0	0.2684	$\sqrt(x_1)(x_0 + (-x_2 x_3 + 0.597)\sin(x_4 + \sin(x_1)))$
18.0	0.1995	$-x_0\sqrt(x_1)((\sin(x_3) - 0.641)\exp(x_4)\cos(x_3) - 1)$
25.0	0.1904	$x_0(\sqrt(x_1) + (-x_3 + 0.715)(\sin(x_1) + \sin(x_2 x_4))\cos(x_1))$

Table 3.4: Solar power study, method GP: formulae of the Pareto front models for seed 42.

From Table 3.4 we see that GP follows a very reasonable strategy: First, it recognizes that the persistence method is a legitimate thing, with production tomorrow being the same as today ($x_1 = P(t)$). Up to a complexity of 5, the identified models only depend on the solar power x_1 and describe with increasing accuracy the conditioned average daily profile. The more complex models include the weather data and forecast. The geometric mean of current power and predicted temperature is present. However, due to the low quality weather forecast as well as the seasonal weather difference between training and testing data, there is no net gain in prediction quality.

Without any further analysis, the model with the lowest testing error is chosen. In Fig. 3.13 (a) we confront the real time series with the prediction from GP for the model of complexity 4. One clearly finds the conditioned average profile. This predicts the production onset a bit too early. The error distribution is shown in Fig. 3.13 (b), where we recognize an asymmetric error distribution with more probable under- than overprediction.

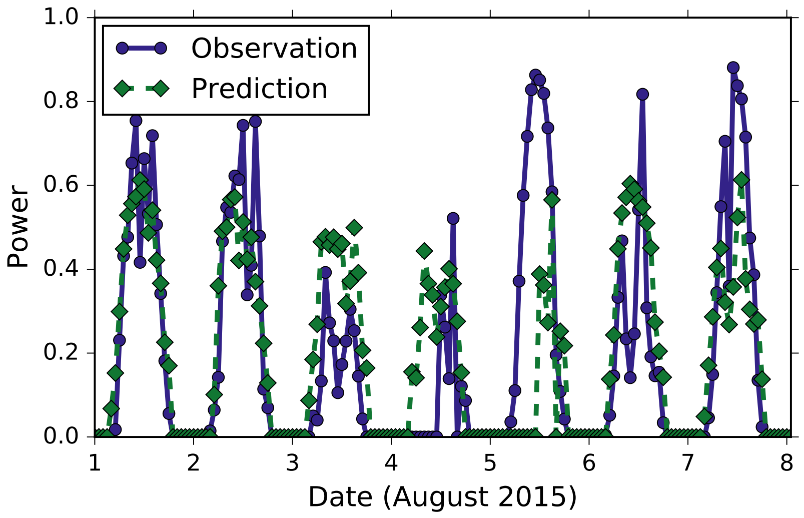
FFX Results

The results of the FFX method are shown in Fig. 3.14-Fig. 3.15 and the models in Table 3.5. As shown, the FFX method is less capable of predicting longer inactive periods, such as at night, where no solar power is produced. This is clearly visible in Fig. 3.15. One may wonder why a constant (complexity zero) explains about 30% of the data (Fig. 3.14). To understand this, let us consider the example of a uniform distribution: the standard deviation is 0.366 ($\frac{1}{\sqrt{12}}$), for a Gaussian distribution the standard deviation from the mean is 0.34 (one-sided). This coincides with our numbers, where we note that a Gaussian is only illustrative, since the power is strictly nonnegative.

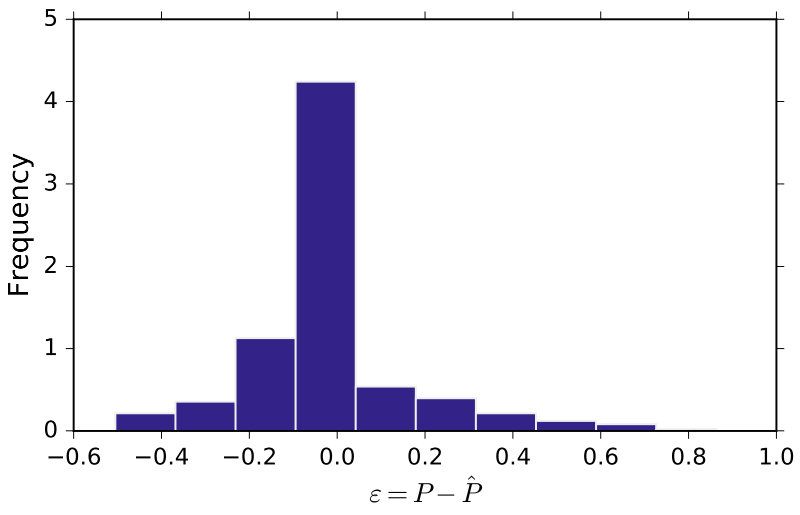
Analyzing the equations of Table 3.5, we notice that the best FFX function is a quadratic form with maxima to limit the signal above zero. This amounts to recover the mean shape of the signal as a quadratic function. Unfortunately this seems almost trivial since one could obtain this mean shape by purely geometrical considerations with a factor for the cloud coverage.

Complexity Γ_2	Error Γ_1	Formula
0	0.2694	0.221
1	0.1996	$0.108 + 0.511x_1$
2	0.1941	$0.0223 + 0.606x_1 + 0.139x_0$
3	0.1934	$0.0470 + 0.436x_1 + 0.328x_0x_1 + 0.138x_4x_0$
4	0.1899	$0.459 - 0.458\max(0, 0.200 - x_1) - 0.339\max(0, 0.333 - x_1) - 0.134\max(0, 0.733 - x_1) - 0.0828\max(0, 0.867 - x_1)$
5	0.1814	$0.301 - 1.25\max(0, 0.333 - x_1)\max(0, 0.200 - x_1) - 0.810\max(0, 0.467 - x_1)\max(0, 0.200 - x_1) + 0.457x_0x_1 - 0.252\max(0, 0.333 - x_1) - 0.0794\max(0, 0.200 - x_1)$

Table 3.5: Solar power study, method FFX: formulae of the Pareto front models.



(a)



(b)

Figure 3.13: Solar power study, method GP: a) Real and predicted time series. We display the results of the first week of August 2015. Prediction used model of complexity 4 which had lowest error on the test set. b) Histogram of the residuals $\varepsilon = P - \hat{P}$. The distribution is asymmetric around zero. The model tends to underpredict.

Summarizing the results for the solar power curves, both methods are able to reproduce the true curve to approximately 20% which is legitimate for a nonoptimized method. The detection of changes when clear sky switches to partially or fully clouded one is not entirely satisfactory and one needs to investigate the improvement of weather predictions for a single location. As said in the introduction, a perfect weather prediction with high resolution would render this work useless for power production forecast (although not for other questions).

Nevertheless, we note that the results in the form of analytic models are highly valuable, because interpretations and further mathematical analysis are possible.

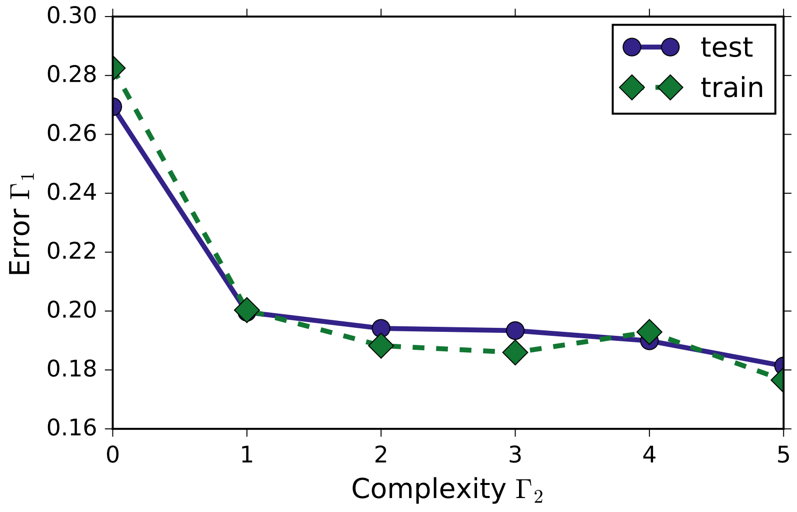
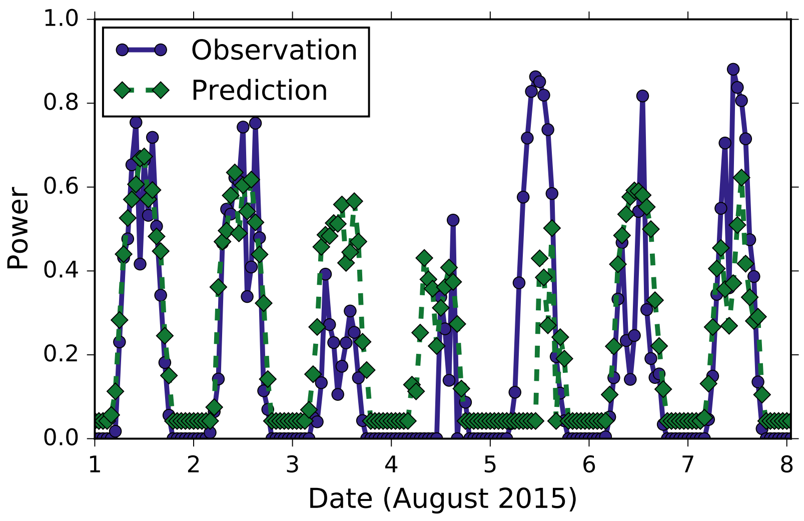
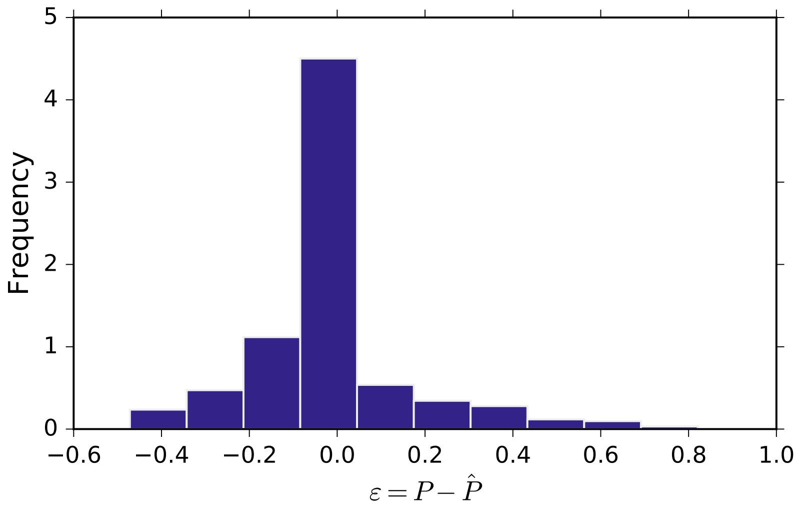


Figure 3.14: Solar power study, Pareto front obtained using FFX. The results for FFX are as accurate as the ones obtained with GP. Test and training set are, however, nicely aligned. This demonstrates not only consistency of the models, but less variability of the models found.



(a)



(b)

Figure 3.15: Solar power study, method FFX: a) Timeseries of the predicted ($\hat{P}$), and observed (P) data. We display the results of the first week of August 2015. Similar to the GP prediction extrema are not particularly well predicted. For the linear model, even the zero values are not well hit. The reason for this is the regression to mean values and the inability of powers to stay at zero for a sufficient time. b) Histogram of the residuals $\varepsilon = P - \hat{P}$. Despite different formulas, the histogram of the residuals is asymmetric around zero with a trend to underpredict as well.

3.5 CONCLUSION

We have demonstrated the use of symbolic regression combined with complexity analysis of the resulting models for the prediction of dynamical systems. More precisely, we identify a system of equations yielding optimal forecasts in terms of a minimized normalized root mean squared error of the difference between model forecast and observation of the system state. We did not investigate theoretical aspects such as the underlying state space. These will be subject of future investigations. Such work is to be carried out carefully to find the limitations of the approach, in particular of genetic programming, which is rather uncontrolled in the way the search space is explored. On the other hand, the methods stand in line with a large collection of methods from regression and classification and one can use much of this previous knowledge. In our opinion, the multiobjective analysis is crucial to identify models to a degree such that they can be used in practice. Probably, this approach will prove very helpful if used in combination with scale analysis, e. g. by prefiltering the data on a selected spatio-temporal scale and then identify equations for this level.

We have tried to show the capabilities of symbolic regression by three examples of increasing complexity: a trivial one - the harmonic oscillator with an almost perfect predictive power, a collection of excitable oscillators where we demonstrated that the methods can perform a kind of multi-scale analysis based on the data. Thirdly, examining the one-day-ahead forecasting of solar power production we have shown that even for messy data we can successfully apply symbolic regression. We expect symbolic regression to outperform classical methods by a few percent in NRMSE. For theoretical considerations, this might be negligible, for real world applications, a few percent might translate into a considerable advantage, since the usage of rare resources can be optimized.

A question for further research is how we can use simplification during the GP iteration to alter the complexity. It may be even a viable choice to control the complexity growth over time, the so-called bloat, in single objective genetic programming - a topic of ongoing interest [Q49]. Additionally, we introduced an intermediate step to only allow for one of many identical solutions for further evolution. One could consider to expand the idea of identical expression trees to include symmetries.

We conclude that symbolic regression is very useful for the prediction of dynamical systems, based on observations only. Our future research will focus on the use of equations to couple the systems to other macroscopic ones (e. g. finance, in the case of wind power), and on the analysis of system stability and other fundamental properties using the found equations, which is scientifically a very crucial point.

ACKNOWLEDGEMENTS

We acknowledge discussion with C. Cornaro and M. Pierro on solar power forecast, helpful words on machine learning by M. Segond. We thank the unisolar initiative for their provision of the data. M. Quade and M. Abel acknowledge support by the German ministry of economy, ZIM project "green energy forecast", grant ID KF2768302ED4.

APPENDIX: HARMONIC OSCILLATOR NOISE STUDY WITH GP

An analogous case study as presented in Section 3.4.1 using GP requires substantial code development and computational time, and is subject of ongoing research. For completeness, we depict a small case study in Fig. 3.16.

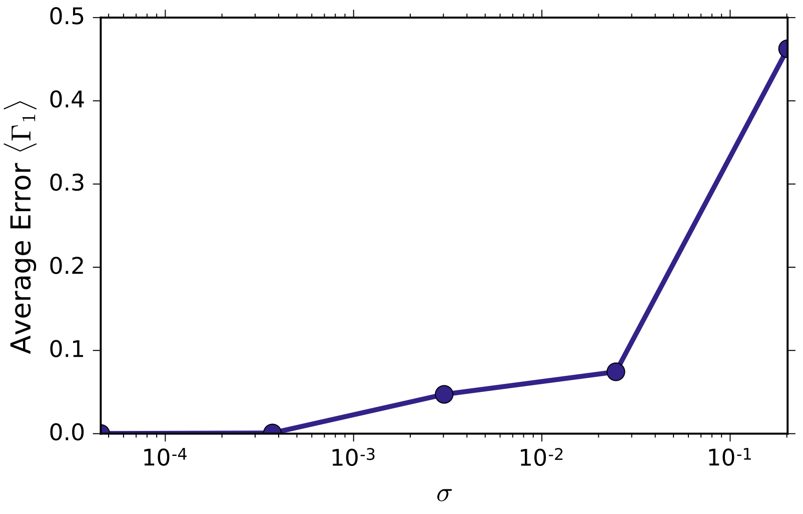


Figure 3.16: Harmonic oscillator study, method GP: NRMSE Equation 3.6 versus noise level σ. We fixed $N = 50$ and $\tau = 10$. The results are averaged 30 times. The absolute values for the NRMSE are higher compared to the results obtained with FFX (Fig. 3.16).

REFERENCES

[Q1] Edward S. Sarachik and Mark A. Cane. *The El Niño–Southern Oscillation Phenomenon*. Cambridge Books Online. Cambridge University Press, 2009. DOI: 10.1017/cbo9780511817496.

[Q2] Carl Friedrich Gauss. *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium*. Cambridge University Press, 2009. DOI: 10.1017/cbo9780511841705.

[Q3] E. Schrödinger. “An Undulatory Theory of the Mechanics of Atoms and Molecules.” In: *Phys. Rev.* 28.6 (Dec. 1926), pp. 1049–1070. DOI: 10.1103/physrev.28.1049.

[Q4] Luca Cardelli and Peter Wegner. “On understanding types, data abstraction, and polymorphism.” In: *CSUR* 17.4 (Dec. 1985), pp. 471–523. DOI: 10.1145/6041.6042.

[Q5] Henning U. Voss, Paul Kolodner, Markus Abel, and Jürgen Kurths. “Amplitude Equations from Spatiotemporal Binary-Fluid Convection Data.” In: *Phys. Rev. Lett.* 83.17 (Oct. 1999), pp. 3422–3425. DOI: 10.1103/physrevlett.83.3422.

[Q6] Markus Abel, Karsten Ahnert, Jürgen Kurths, and Simon Mandelj. “Additive nonparametric reconstruction of dynamical systems from time series.” In: *Phys. Rev. E* 71.1 (Jan. 2005), p. 015203. DOI: 10.1103/physreve.71.015203.

[Q7] Markus Abel. “Nonparametric Modeling And Spatiotemporal Dynamical Systems.” In: *Int. J. Bifurcation Chaos* 14.06 (June 2004), pp. 2027–2039. DOI: 10.1142/s0218127404010382.

[Q8] Daniel Rey, Michael Eldridge, Mark Kostuk, Henry D.I. Abarbanel, Jan Schumann-Bischoff, and Ulrich Parlitz. "Accurate state and parameter estimation in nonlinear systems with sparse observations." In: *Phys. Lett. A* 378.11-12 (Feb. 2014), pp. 869–873. DOI: 10.1016/j.physleta.2014.01.027.

[Q9] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems." In: *Proc Natl Acad Sci USA* 113.15 (Mar. 2016), pp. 3932–3937. DOI: 10.1073/pnas.1517384113.

[Q10] F. Takens. *Detecting strange attractors in turbulence*. Lecture Notes in Mathematics No.898. Berlin: Springer, 1981.

[Q11] Tim Sauer, James A. Yorke, and Martin Casdagli. "Embedology." In: *J Stat Phys* 65.3-4 (Nov. 1991), pp. 579–616. DOI: 10.1007/bf01053745.

[Q12] Karsten Ahnert and Markus Abel. "Numerical differentiation of experimental data: Local versus global methods." In: *Comput. Phys. Commun.* 177.10 (Nov. 2007), pp. 764–774. DOI: 10.1016/j.cpc.2007.03.009.

[Q13] J. B. Tenenbaum. "A Global Geometric Framework for Nonlinear Dimensionality Reduction." In: *Science* 290.5500 (Dec. 2000), pp. 2319–2323. DOI: 10.1126/science.290.5500.2319.

[Q14] A. Singer, R. Erban, I. G. Kevrekidis, and R. R. Coifman. "Detecting intrinsic slow variables in stochastic dynamical systems by anisotropic diffusion maps." In: *Proceedings of the National Academy of Sciences* 106.38 (Aug. 2009), pp. 16090–16095. DOI: 10.1073/pnas.0905547106.

[Q15] S. T. Roweis. "Nonlinear Dimensionality Reduction by Locally Linear Embedding." In: *Science* 290.5500 (Dec. 2000), pp. 2323–2326. DOI: 10.1126/science.290.5500.2323.

[Q16] Michael Schmidt and Hod Lipson. "Distilling Free-Form Natural Laws from Experimental Data." In: *Science* 324.5923 (Apr. 2009), pp. 81–85. DOI: 10.1126/science.1165893.

[Q17] J. Bongard and H. Lipson. "Automated reverse engineering of nonlinear dynamical systems." In: *Proceedings of the National Academy of Sciences* 104.24 (June 2007), pp. 9943–9948. DOI: 10.1073/pnas.0609476104.

[Q18] Michael D Schmidt, Ravishankar R Vallabhajosyula, Jerry W Jenkins, Jonathan E Hood, Abhishek S Soni, John P Wikswo, and Hod Lipson. "Automated refinement and inference of analytical models for metabolic networks." In: *Phys. Biol.* 8.5 (Aug. 2011), p. 055011. DOI: 10.1088/1478-3975/8/5/055011.

[Q19] K. Rodríguez-Vázquez and P.J. Fleming. "Multi-objective genetic programming for nonlinear system identification." In: *Electron. Lett.* 34.9 (1998), p. 930. DOI: 10.1049/el:19980632.

[Q20] N. Gautier, J.-L. Aider, T. Duriez, B. R. Noack, M. Segond, and M. Abel. "Closed-loop separation control using machine~learning." In: *J. Fluid Mech.* 770 (Apr. 2015), pp. 442–457. DOI: 10.1017/jfm.2015.95.

[Q21] Steven L. Brunton and Bernd R. Noack. "Closed-Loop Turbulence Control: Progress and Challenges." In: *Appl. Mech. Rev* 67.5 (Aug. 2015), p. 050801. DOI: 10.1115/1.4031175.

[Q22] Trent McConaghy. "FFX: Fast, Scalable, Deterministic Symbolic Regression Technology." In: *Genetic and Evolutionary Computation*. Springer New York, 2011, pp. 235–260. DOI: 10.1007/978-1-4614-1770-5_13.

[Q23] François-Michel De Rainville, Félix-Antoine Fortin, Marc-André Gardner, Marc Parizeau, and Christian Gagné. "Deap. enabling nimbler evolutions." In: *SIGEVOlution* 6.2 (Feb. 2014), pp. 17–26. DOI: 10.1145/2597453.2597455.

[Q24] SymPy Development Team. *SymPy: Python library for symbolic mathematics*. 2016. URL: %7Bhttp://www.sympy.org%7D.

[Q25] E. Lorenzo. *Solar Electricity: Engineering of Photovoltaic Systems*. PROGENSA, 1994.

[Q26] Edward N. Lorenz. "Deterministic Nonperiodic Flow." In: *J. Atmos. Sci.* 20.2 (Mar. 1963), pp. 130–141. DOI: 10.1175/1520-0469(1963)020<0130:dnf>2.0.co;2.

[Q27] Holger Kantz and Thomas Schreiber. *Nonlinear Time Series Analysis*. Cambridge University Press, 2003. DOI: 10.1017/cbo9780511755798.

[Q28] Kathleen T. Alligood, Tim D. Sauer, and James A. Yorke. *Chaos*. Series in Nonlinear Science. New York: Springer Berlin Heidelberg, 1997. DOI: 10.1007/978-3-642-59281-2.

[Q29] Steven H Strogatz. *Nonlinear dynamics and chaos: With applications to physics, biology, chemistry, and engineering*. Westview press, 2014.

[Q30] Hassler Whitney. "Differentiable Manifolds." In: *The Annals of Mathematics* 37.3 (July 1936), p. 645. DOI: 10.2307/1968482.

[Q31] *Accuracy (trueness and precision) of measurement methods and results — Part 1: General principles and definitions*. Standard. Geneva, CH: International Organization for Standardization, 1994.

[Q32] Henry D. I. Abarbanel. *Analysis of Observed Chaotic Data*. New York: Springer New York, 1996. DOI: 10.1007/978-1-4612-0763-4.

[Q33] N.A. Gershenfeld. *The Nature of Mathematical Modeling*. Cambridge University Press, 1999.

[Q34] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer US, 1971. DOI: 10.1007/978-1-4615-7566-5.

[Q35] G.B. Dantzig, R. Cottle, and Y.P. Aneja. *Mathematical programming: Essays in honor of George B. Dantzig*. Mathematical Programming: Essays in Honor of George B. Dantzig Bd. 1. North-Holland, 1985.

[Q36] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[Q37] Hui Zou and Trevor Hastie. "Regularization and variable selection via the elastic net." In: *J Royal Statistical Soc B* 67.2 (Apr. 2005), pp. 301–320. DOI: 10.1111/j.1467-9868.2005.00503.x.

[Q38] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. "Occam's Razor." In: *Information Processing Letters* 24.6 (Apr. 1987), pp. 377–380. DOI: 10.1016/0020-0190(87)90114-1.

[Q39] William H Jefferys and James O Berger. "Ockham's razor and Bayesian analysis." In: *Am. Sci.* (1992), pp. 64–72.

[Q40] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. "A fast and elitist multiobjective genetic algorithm: Nsga-ii." In: *IEEE Trans. Evol. Computat.* 6.2 (Apr. 2002), pp. 182–197. DOI: 10.1109/4235.996017.

[Q41] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. "SPEA2: Improving the strength Pareto evolutionary algorithm." In: *TIK-Report*. Vol. 103. Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK), 2001. DOI: 10.3929/ethz-a-004284029.

[Q42] J. Knowles and D. Corne. "The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation." In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. IEEE. DOI: 10.1109/cec.1999.781913.

[Q43] Guido F. Smits and Mark Kotanchek. "Pareto-Front Exploitation in Symbolic Regression." In: *Genetic Programming Theory and Practice II*. Springer-Verlag, 2005, pp. 283–299. DOI: 10.1007/0-387-23254-0_17.

[Q44] E.J. Vladislavleva, G.F. Smits, and D. den Hertog. "Order of Nonlinearity as a Complexity Measure for Models Generated by Symbolic Regression via Pareto Genetic Programming." In: *IEEE Trans. Evol. Computat.* 13.2 (Apr. 2009), pp. 333–349. DOI: 10.1109/tevc.2008.926486.

[Q45] Unisolar Potsdam e.V., T. Felbinger. *Unisolar - Energy Sans Souci*. URL: http://www.unisolar-potsdam.de (visited on 04/29/2018).

[Q46] J. Nagumo, S. Arimoto, and S. Yoshizawa. "An Active Pulse Transmission Line Simulating Nerve Axon." In: *Proc. IRE* 50.10 (Oct. 1962), pp. 2061–2070. DOI: 10.1109/jrproc.1962.288235.

[Q47] A. L. Hodgkin and A. F. Huxley. "A quantitative description of membrane current and its application to conduction and excitation in nerve." In: *The Journal of Physiology* 117.4 (Aug. 1952), pp. 500–544. DOI: 10.1113/jphysiol.1952.sp004764.

[Q48] Ecmwf. *IFS Documentation CY41R1*. IFS Documentation. Reading, England: ECMWF, 2015.

[Q49] Marc-André Gardner, Christian Gagné, and Marc Parizeau. "Controlling code growth by dynamically shaping the genotype size distribution." In: *Genet Program Evolvable Mach* 16.4 (Feb. 2015), pp. 455–498. DOI: 10.1007/s10710-015-9242-8.

4 SYNCHRONIZATION CONTROL OF OSCILLATOR NETWORKS USING SYMBOLIC REGRESSION

by Julien Gout[1, 2], Markus Quade[1, 2], Kamran Shafi[3], Robert K. Niven[3], Markus Abel[3]

[1] Universität Potsdam, Institut für Physik und Astronomie, Karl-Liebknecht-Straße 24/25, 14476 Potsdam, Germany
[2] Ambrosys GmbH, David-Gilly Straße 1, 14469 Potsdam, Germany
[3] School of Engineering and Information Technology, The University of New South Wales, Canberra ACT 2600, Australia

This paper has been publish in Nonlinear Dynamics. We investigate Synchronization in coupled FitzHugh and show how to control these oscillators using GP*-based symbolic regression.*

Abstract

Networks of coupled dynamical systems provide a powerful way to model systems with enormously complex dynamics, such as the human brain. Control of synchronization in such networked systems has far-reaching applications in many domains, including engineering and medicine. In this paper, we formulate the synchronization control in dynamical systems as an optimization problem and present a multi-objective genetic programming-based approach to infer optimal control functions that drive the system from a synchronized to a non-synchronized state and vice versa. The genetic programming-based controller allows learning optimal control functions in an interpretable symbolic form. The effectiveness of the proposed approach is demonstrated in controlling synchronization in coupled oscillator systems linked in networks of increasing order complexity, ranging from a simple coupled oscillator system to a hierarchical network of coupled oscillators. The results show that the proposed method can learn highly effective and interpretable control functions for such systems.

Keywords— Dynamical Systems, Synchronization Control, Genetic programming

4.1 INTRODUCTION

The control of dynamical systems lies at the heart of modern engineering [R1], and in many other disciplines, including physics [R2, R3] and medicine [R4, R5]. This paper specifically focuses on the control of synchronization in dynamical systems. Synchronization is a widespread phenomenon observed in many natural and engineered complex systems whereby locally interacting components of a complex system tend to coordinate and exhibit collective behavior [R6, R7]. In dynamical systems, synchronization refers to the coordination phenomenon between multiple weakly coupled independent oscillating systems that influences the overall dynamics of the system. The role of synchronization control is to moderate this behavior (e. g. , to drive the system into or out of synchronization) by applying an external force or control signal [R6]. Synchronization control has significant implications for numerous application domains in engineering and science, including communications [R8], teleoperations [R9, R10] and brain modeling [R11], to name a few. A more specialized overview of synchronization in oscillators, and especially phase oscillators, can be found in [R12].

Several approaches exist for the control of dynamical systems, such as those based on control theory [R13], mathematical and numerical optimization [R14] and computational intelligence [R15] techniques. The "optimal control" methods [R16], in particular, aim at driving and maintaining a dynamical system in a desired state. This is generally achieved by finding a control law, in the form of a set of differential equations, which optimizes (by maximizing or minimizing) a cost function related to the control task. For instance, in a medical application, the control of body tremors (e. g. , due to seizures) may be achieved by minimizing the amplitude of body oscillations.

If the system is known in terms of a mathematical description, linear theory can be used [R17, R18] in many cases to find the optimal control. However, for nonlinear, extended and consequently complex systems, linear theory may fail. In such cases, more general methods are needed to learn effective control laws. We have been looking for the most general method using analytical expressions, or in other words a method to infer control laws from an arbitrary domain which can be defined in a general way. We identified evolutionary machine learning methods as a suitable source of algorithms. Specifically, we describe the application of Genetic Programming (GP) [R19] to control synchronization in coupled networks, including a hierarchical network of coupled oscillators. Unlike neural networks and other black-box artificial intelligence methods that are commonly applied to optimal control, GP allows dynamically learning complex control laws in an interpretable symbolic form — a method that is referred as symbolic regression [R20, R21, R22]. Previously, evolutionary algorithms have been successfully used to optimize parameters for model-based control of synchronization [R10, R23]. In contrast, we optimize the full expression, and not only parameters.

Previous attempts to use symbolic regression for the control of dynamical systems were mostly restricted to experiments [R1, R24, R25] without multi-objectivity and without the optimization of constants involved in the equations. In contrast, here we use a multi-objective formulation of GP, which allows learning much sparser, as well as multiple Pareto (or non-dominated) solutions [R20, R21, R22]. The Pareto solutions can be further analyzed for insight using the conventional analytical methods, such as bifurcation analysis — subject of our ongoing work. We demonstrate the effectiveness of the proposed control approach through application to different dynamical systems with growing level of complexity, ranging from a single oscillator to a hierarchical network. For each system, we show the useful optimal control terms found by symbolic regression to synchronize or desynchronize the oscillators. The application to the control of synchronization in a hierarchical network of oscillators is motivated by brain disorder problems in the medical domain. Body tremors occur when firing neurons synchronize in regions of brain [R4]. In a normal brain state, neurons are coupled to neighboring neurons such that adjacent neurons influence mutually. If the firing is periodic, which may appear due to the inherent dynamics of the excitable neurons, this mutual influence may give rise to synchronization [R6, R26]. If the coupling term is very large, this synchronization may extend over a whole region in our brain and thus over many neurons. Eventually this collective firing leads to shaky movements of hands, arms or the head, and is treated as a brain disorder. One remedy to this problem is to implant a control device which resets the neurons and counteracts the collective synchronization. An evident question then is how to design such a controller which also minimizes design cost, energy consumption, or other medical constraints. We model this phenomenon, in this work, as a set of oscillatory units (representing neurons) coupled in a small hierarchical network, and apply the proposed approach for learning the optimal control laws for this model.

To the best of our knowledge, this is the is first application of a multi-objective GP to synchronization control in networks of coupled dynamical systems. The rest of the paper is organized as follows: Section 4.2 provides the background information on control of dynamical systems, the common approaches for optimal control of dynamical systems, GP, and the specific methods used in our implementation of GP. Fig. 4.2.2 provides the details of the common GP parameters and other experimental settings used to evaluate the proposed method. The specific parameter settings are provided in each corresponding section. As a proof of concept, Section 4.3 demonstrates the application of GP-based control using two simple benchmark dynamical system examples: a harmonic oscillator and the Lorenz system. GP is used to learn control to bring these system into a chaotic state or back. Section 4.4 presents our study of GP application to networked dynamic systems. Four systems are tested in this section, including a simple coupled oscillator system; and three systems of oscillators coupled, respectively, in a one-dimensional ring network; a two-dimensional torus network; and a hierarchical network. The paper concludes in Section 4.5.

4.2 METHODS

4.2.1 *Optimal control of dynamical systems*

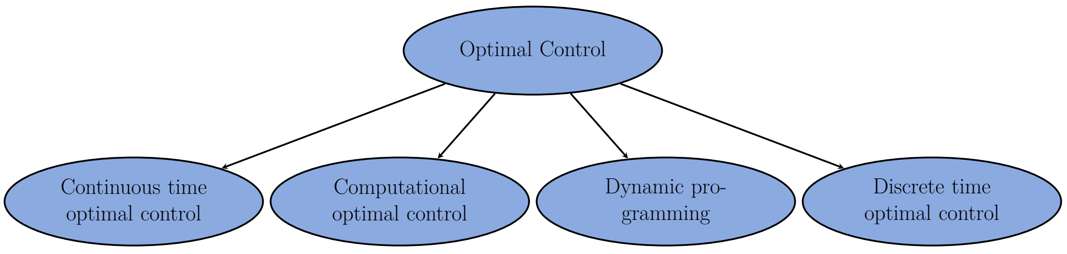

Optimal Control
Continuous time optimal control
Computational optimal control
Dynamic programming
Discrete time optimal control

Figure 4.1: An overview of different types of optimal control.

The control of a dynamical system involves determining and manipulating the trajectory of the system in phase space in order to drive the system to a desired state. The control problem can be formulated as an optimization task with the objective to minimize a cost function defined in terms of the deviation of the state of the system from its desired one. In general, a formal definition of an optimal control problem requires a mathematical model of the system, a cost function or performance index, a specification of boundary conditions on states, and additional constraints.

If there are no path constraints on the states or the control variables, and if the initial and final conditions are fixed, a fairly general continuous time optimal control problem reads: Find the control vector $\vec{u} : \mathbb{R}^{n_x} \times [t_s, t_f] \mapsto \mathbb{R}^{n_u}$ that minimizes the cost function

$$\Gamma = \varphi(\vec{x}(t_f)) + \int_{t_s}^{t_f} L(\vec{x}(t), \vec{u}(\vec{x}, t), t)\mathrm{d}t, \tag{4.1}$$

subject to

$$\dot{\vec{x}} = \vec{f}(\vec{x}, \vec{u}, t), \ \vec{x}(t_s) = \vec{x}_s, \tag{4.2}$$

where $[t_s, t_f]$ is the time interval of interest; $\vec{x}$:$[t_s, t_f] \mapsto \mathbb{R}^{n_x}$ is the state vector; $\varphi : \mathbb{R}^{n_x} \mapsto \mathbb{R}$ is a terminal cost function; L:$\mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \mapsto \mathbb{R}$ is an intermediate cost function; and $\vec{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \mapsto \mathbb{R}^{n_x}$ is a vector field. Note that Eq. (4.2) represents the dynamics of the system and its initial state. This problem definition is known as the Bolza problem; and for $\varphi(x(t_f)) = 0$ and $\vec{u} = \dot{\vec{x}}(t)$ it is known as the Lagrange problem [R27]. Also note that the performance index Γ is a functional, which is used to assign a real value to each control function $\vec{u}$ in a class.

The solutions to many optimal control problems cannot be found by analytical means. Over the years, many computational methods have been developed to solve general optimal control problems. The choice of a method for addressing an optimal control problem may depend on a number of factors, including the types of cost functions, time domain, and constraints considered in the problem. Fig. 4.1 shows different methods used in the optimal control of dynamical systems. Among these methods, the direct methods work by discretizing the control problem and solving it using nonlinear programming approaches. Some methods involve the discretization of the differential equations by defining a grid of N points covering the time interval $[t_s, t_f]$, $t_s = t_1 < t_2 < \ldots < t_N = t_f$, and solving these equations using, for instance, Euler, trapezoidal, or Runge–Kutta methods [R28]. In this approach, the differential equations become equality constraints of the nonlinear programming problem. Other direct methods involve the approximation of control and states using basis functions, such as splines or Lagrange polynomials.

The continuous-time problems mentioned above have discrete time counterparts. These formulations are useful when the dynamics are discrete (for example, a multistage system), or when dealing with computer controlled systems. In discrete time, the dynamics can be expressed as a difference equation:

$$\vec{x}(k+1) = \vec{f}(\vec{x}(k), \vec{u}(k), k), \ \vec{x}(N_0) = \vec{x}_s,$$

where k is an integer index, $\vec{x}(k)$ is the state vector, $\vec{u}(k)$ is the control vector, and $\vec{f}$ is a vector function. The objective is to find a control sequence $\vec{u}(k)$, $k = N_0, \dots, N_f - 1$, to minimize a performance index of the form:

$$\Gamma = \varphi(\vec{x}(N_f)) + \sum_{k=N_0}^{N_f-1} L(\vec{x}(k), \vec{u}(k), k).$$

See, for example, [R29, R30] for further details on discrete-time optimal control.

Dynamic programming is an alternative to the variational approach to optimal control. It was proposed by Bellman in the 1950s and is an extension of Hamilton–Jacobi theory. A number of books exist on these topics including [R29, R17, R30]. A general overview of the optimal control methods for dynamical systems can be found in [R16]. For further details, readers are referred to [R31].

Unless otherwise stated, we approach the control problems presented in this paper using discrete-time numerical methods. However, for most purposes, the continuous-time formulation given in Eqs. (4.1) and (4.2) can be adopted unchanged for our control methods. One major generalization, though, will be made in the context of multi-objective optimization (Fig. 4.2.2); there Γ is replaced by a vector of independent cost functionals $\vec{\Gamma} = (\Gamma_1, \dots, \Gamma_N)$. Another adjustment concerns the specialization of $\vec{f}$ and $\vec{u}$ for the particular control scheme considered here, as will be described next.

A feedback control scheme [R32] is adopted in this work to implement the control. Fig. 4.2 depicts the general architecture.

For this particular scheme Eq. (4.2) can be rewritten as:

$$\dot{\vec{x}} = \vec{f}(\vec{x}, t) + \vec{a}, \;\; \vec{x}(t_s) = \vec{x}_s,$$

where the uncontrolled system $\dot{\vec{x}} = \vec{f}(\vec{x}, t)$ is controlled by an additive actuator term $\vec{a}$, and the control function now depends on sensor measurements, given by the output vector $\vec{s} \in \mathbb{R}^{n_s}$:

$$\vec{a} = \vec{u}(\vec{s}, t).$$

These measurements might be nonlinear functions of the state $\vec{x}$. For simplicity, external perturbations to the dynamic system are not considered here.

4.2.2 *Genetic Programming*

Genetic Programming (GP) [R19, R33] is an evolutionary algorithm in the class of meta-heuristic search techniques that promise global optimization. Similar to the Genetic Algorithm (GA), GP also uses the natural selection metaphor to evolve a set, or so-called population, of solutions or individuals using a cost-based selection mechanism. The evolution occurs over a number of iterations, called generations. GP differs from GA mainly in the representation of a solution. A solution in GP is generally represented using lists or expression trees. Expression trees are constructed from the elements of two predefined primitive sets: a function set consisting of mathematical operators and trigonometric functions, such as {`+`, `-`, `*`, `cos`, `sin`}, and a terminal set consisting of variables and constants, such as { `x`,

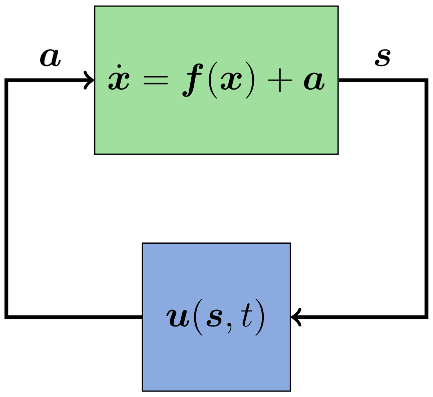

$\boldsymbol{a}$
$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{a}$
$\boldsymbol{s}$
$\boldsymbol{u}(\boldsymbol{s}, t)$

Figure 4.2: Sketch of the feedback control loop. The output of the dynamical system $\vec{x}$ is measured by sensors $\vec{s}$ which are used as input to the control function $\vec{u}$. The control function, in turn, acts on the system via actuators $\vec{a} = \vec{u}(\vec{s}, t)$ in order to achieve a desired state. (External disturbances which can be incorporated explicitly as additional inputs to the dynamical system and the control function are not shown here.)

`y, b`}. Function symbols make up the internal nodes of a tree; and terminal symbols are used in the leaf nodes. For example, Fig. 4.3 shows the tree representation for the expression $b \cdot x + \cos(y)$. All elements of the tree are drawn from the aforementioned primitive sets: the variables and constants in the terminal set (x, y, and b) form the leaves of the tree and the mathematical symbols in the functional set ($\cdot$, $+$, and cos) are used in forming the tree's internal nodes.

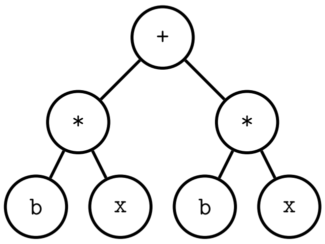


Figure 4.3: Tree representation of the mathematical expression $b \cdot x + \cos(y)$. The symbols `b`, `x`, and `y` are taken from the terminal set and make up the leaf nodes of the tree, whereas the symbols `*`, `+`, and `cos`, are symbols taken from the function set, they make up the internal nodes.

Algorithm 2 Top level description of a GP algorithm

```
procedure MAIN
    $G_0 \leftarrow \text{random}(\lambda)$
    $\text{evaluate}(G_0)$
    $t \leftarrow 1$
    repeat
        $O_t \leftarrow \text{breed}(G_{t-1}, \lambda)$
        $\text{evaluate}(O_t)$
        $G_t \leftarrow \text{select}(O_t, G_{t-1}, \mu)$
        $t \leftarrow t + 1$
    until $t > T$ or $G_t = \text{good}()$
end procedure
```

A standard GP algorithm, see 2, begins by generating a population of random solutions G_0. A random variable length solution, with a given maximum tree depth, is generated by choosing operators, functions and variables from the two sets uniform randomly. Each solution is then evaluated in a given task, e. g. , learning underlying relationship between a given set of variables. A cost is assigned to each solution based on its performance in solving the task, e. g. how closely a solution predicted the target function output. A new population of solutions O_t is then generated by: (i) probabilistically selecting parent solutions from the existing population using a cost-proportional selection mechanism, and (ii) creating offspring or new solutions by applying recombination (or crossover) and variation (or mutation) operators (see Fig. 4.4). This operation is repeated until a given number of solutions (a fixed population size) is reached. A closure property is always maintained to ensure only valid solutions are generated during both the initialization and breeding operations. An elitist approach is commonly used for improving the algorithm's convergence speed. This involves copying some of the high-performing parent solutions to the next-generation population G_{t+1}. The selection, evaluation and reproduction processes are repeated until a given stopping criteria is met, commonly a fixed number of maximum generations. For further details about GP operation, readers are referred to [R34, R35].

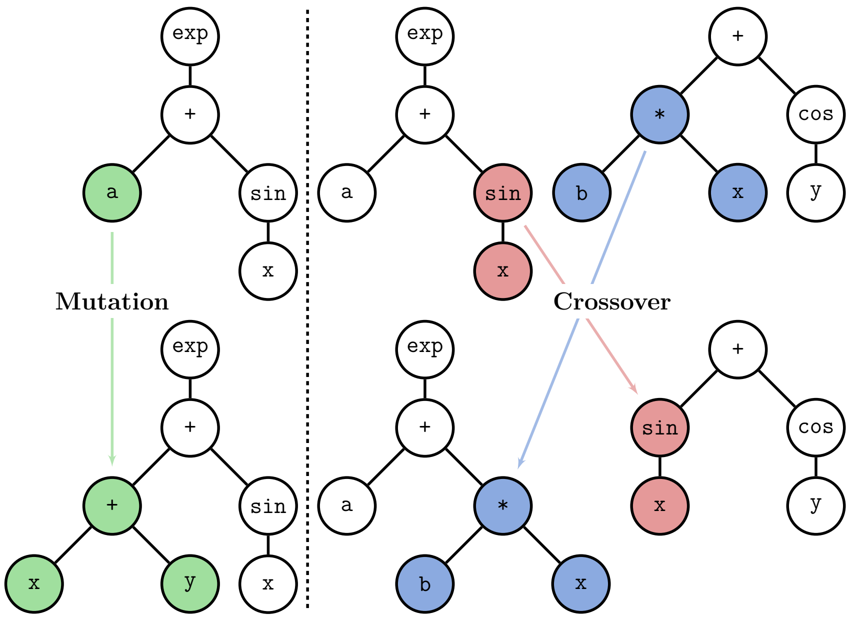


Figure 4.4: Breeding: Mutation and crossover operations on expression trees. *Source:* Adapted from [R22]; used with permission.

The original motivation behind GP was to automatically produce computer programs, similar to a human strategy [R19]. However, since its inception, GP has been applied to various tasks, including the traditional machine learning [R36] and optimization [R15] tasks. In the context of learning dynamical system models or control laws for dynamical systems, GP is a preferred choice for two main reasons: First, the expression tree representation used by GP is human interpretable and clearly provides an edge over the blackbox models learned by other computational approaches, such as artificial neural networks. Second, the GP solutions can be readily represented as mathematical equations and evaluated as control laws for dynamical systems.

The application of GP to a general control problem (*cf.* Section 4.2.1), can be formulated as a learning and optimization task. That is, we would like to learn a control function to drive and keep a dynamical system in a desired state. This requires minimizing a cost function ($\vec{\Gamma}$), e. g. , the difference between a given state in time and the desired state. For most practical purposes, $\vec{\Gamma}$ can be expected to have complex properties including non-linearity, multi-modality, multi-variability and discontinuity. This poses a serious challenge to many traditional direct and gradient methods. In turn, meta-heuristic methods, such as GP, are suitable candidates for this task. Fig. 4.5 depicts how GP-based dynamic controller is used within a feedback control loop, shown in Fig. 4.2.

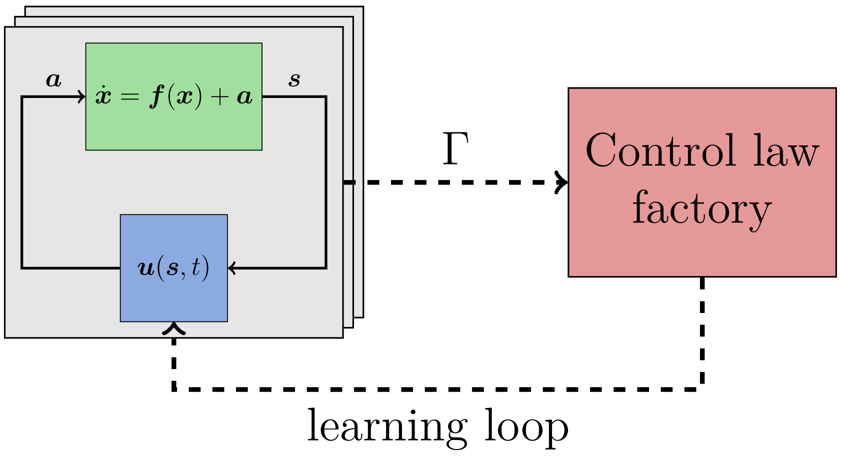


Figure 4.5: Sketch of the machine learning loop. Using an evolutionary metaphor, the GP algorithm generates a set of candidate control solutions $\vec{u}$, called the population. The candidate solutions are then evaluated in many realizations of the control loop; the performance in each iteration is rated via a cost functional Γ and fed back as a cost index into the GP algorithm. The algorithm uses the performance rating to select the best solutions and evolve them into the next generation of candidate solutions. This learning loop repeats until at least one satisfactory control law is found (or other break conditions are met).

Multi-Objective Cost Evaluation

Defining a good cost function is a key process in GP that matter, which determines the quality of a solution in the population. A common method for cost assignment is to map solutions' performances to scalar values within a given range, e. g. , $[0, 1]$. This method simplifies the ranking procedure needed for selection and reproduction processes. However, it also limits the number of performance objectives that can be considered in the cost evaluation. A straight-forward way to address this concern is to apply a weighted sum method that in turn still allows mapping multiple performance objectives to scalar values. This type of methods not only require manual tuning of weights but also hide trade-off details between conflicting objectives. An alternative method to handle conflicting performance objectives in the design of cost functions is to use the concept of Pareto dominance [R37]. According to this principle, a solution x dominates another solution y, if x performs better than y in at least one of the multiple objectives or criteria considered and at least equal or better in all other objectives. This concept provides a convenient mechanism to consider multiple conflicting performance objectives simultaneously in ranking solutions based on their domination score. The solutions with the highest scores form the Pareto or efficient frontier. Fig. 4.6 provides an illustration of this concept. Several very successful Evolutionary Multi-Objective Optimization (EMO) algorithms are based on Pareto dominance [R38, R39, R40].

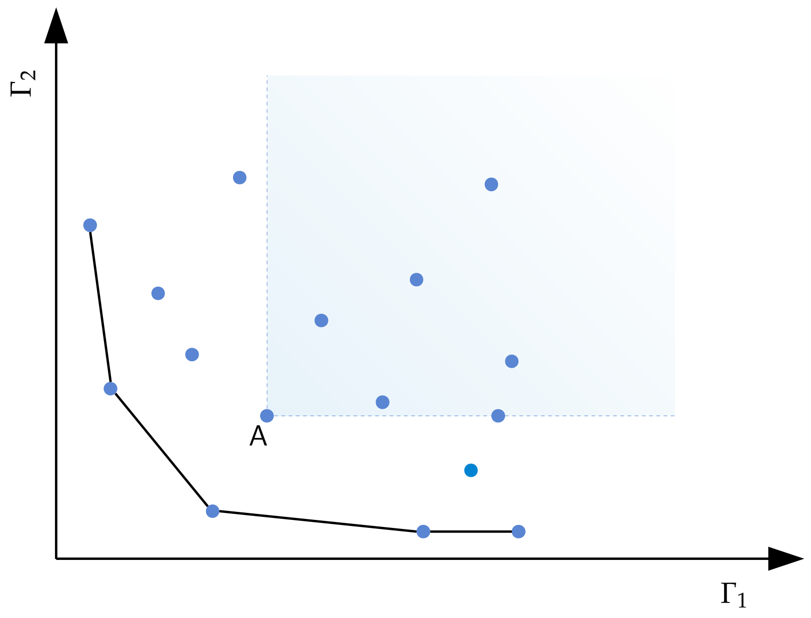


Figure 4.6: Pareto front (blue dots connected by a black line) of a set of candidate solutions evaluated with respect to two cost indexes Γ_1 and Γ_2. The Pareto-optimal solutions are non-dominated with respect to each other but dominate all other solutions. The region marked in light blue illustrates the notion of Pareto dominance: solutions contained within that region are dominated by a solution A.

The standard GP is known to have a tendency to generate long and complicated solutions in order to exactly match, or overfit, performance target (optimal performance in our case). One way to address this issue is to design a cost function where both the performance (e. g. , controller error in our case) and length of solutions are considered explicitly in determining the quality of a solution. Such a multi-objective cost mechanism allows introducing an explicit selection pressure in the evolutionary process and preferring smaller well-performing solutions over their longer counterparts [R22]. Following this line, a multi-objective cost evaluation method is adopted in our implementation of GP in this work. In specific, we adopt the mechanism used in Non-Dominated Sorting Algorithm II (NSGAII) [R38] in our implementation of GP, which combines a non-dominated sorting mechanism with an Euclidean distance-based metric to promote solution diversity and spread or coverage of the entire Pareto front.

Constant Optimization

The learning of numeral constants, if desired, is treated similar to the learning of other terminal variables in the standard GP and an approximation may be learned by sampling over a given range. However, this approach can severely impair the convergence speed of GP as the search space essentially becomes infinite, especially for a continuous representation. Another approach, followed in this work, is to use a traditional parameter optimization algorithm, such as the Levenberg–Marquardt least squares method. By allowing designated symbolic constants $\vec{k} = (k_1, k_2, \ldots)$ in an expression tree, one can determine optimal values for these constants through parameter optimization. Thus, the calculation of the numeral constants becomes another optimization task, with

$$\vec{k}^* = \underset{\vec{k}}{\operatorname{argmin}}\, \Gamma(\vec{u}(\vec{s}, \vec{k})),$$

and

$$\Gamma = \Gamma(\vec{u}(\vec{s}, \vec{k}^*)),$$

effectively introducing two combined layers of optimization.

To incorporate such a regression mechanism, the terminal set can be divided further into an argument set and a constant set, where the argument set contains all the terminal symbols that represent elements from the sensor vector $\vec{s}$, which are passed as arguments to the control function $\vec{u}$. The constant set, on the other hand, consists of all the designated constants representing elements from the constant vector $\vec{k}$. The actual construction of these sets depends on the dynamical system under consideration and the type of the control task.

GP setup

This section gives a brief summary of the specific implementation details and the common parameters used in our experimental setup. Hyperparameters have been chosen empirically such that they lead to plausible and interpretable results on the chosen set of examples. We did not optimize the hyperparameters for convergence.

All computer programs are developed in `Python` using open-source software packages. The implementation of the GP algorithm uses a customized version of the `DEAP` module [R41]. Particularly the routines for tree generation, selection, and breeding were adopted unchanged. Most of the numerical algorithms in use are provided by the `numpy` and `scipy` modules [R42, R43]. Notably, constant optimization is conducted using the Levenberg-Marquardt least squares algorithm (`scipy`) and numerical integration using the dopri5 solver (also `scipy`). Random numbers are generated using the Mersenne Twister pseudo-random number generator provided by the *random* module [R44]. Finally, the `sympy` module is used for the simplification of symbolic mathematical expressions generated from the GP runs [R45]. The code can be found at [R46].

Table 4.1 gives an overview of the methods and parameters used for the GP runs. Actual implementations can be found under the same name in the `DEAP` module.

The function set is chosen such that the operators and functions are defined on $\mathbb{R}$. This allows for an easy evaluation and application of the expressions built

Table 4.1: General setup of the GP runs.

Function set	$\{+, -, \cdot, \sin, \cos, \exp\}$
Population size	500
Max. generations	20
MOO algorithm	NSGA-II
Tree generation	*halfandhalf*
Min. height	1
Max. height	4
Selection	*selTournament*
Tournament size	2
Breeding	*varOr*
Recombination	*cxOnePoint*
Crossover probability	0.5
Crossover max. height	20
Mutation	*mutUniform*
Mutation probability	0.2
Mutation max. height	20
Constant optimization	*leastsq*

from this set and prevents additional handling of singularities. This choice, though, restricts the number of possible solutions. In principle, the inclusion of other functions and operators, such as log, $\sqrt{}$, or $1/x$, is conceivable and would lead to a different space of potential solutions. Where possible, other GP parameter values are chosen according to the best practices used by the GP community; see [R19, R36, R34]. For more involved cases a second stopping criterion is used where the learning is stopped when an error of the order of 10^{-5} is reached. Numerical integration is performed during cost assessment for all of the investigated dynamical systems. As mentioned before, dopri5 is used as solver [R28]: This is an explicit Runge–Kutta method of order (4)5 with adaptive step size. If not otherwise stated, the maximum number of steps allowed during one call is set to 4000, the relative tolerance to 10^{-6}, and the absolute tolerance to 10^{-12}. The pseudorandom number generator is seeded by a randomly selected unique seed for every GP run and never reinitialized during the same run. The specific seed used in an experiment will be explicitly stated in the corresponding setup description. Results obtained from one experiment can thus be duplicated when this same seed is reused in another run of the experiment.

4.3 CONTROL OF INDEPENDENT DYNAMICAL SYSTEMS

In order to establish a baseline, explain the working of the proposed method and determine its effectiveness, in this section we discuss the application of GP-based control methodology to a well-understood example, the harmonic oscillator. It is forced to a fixed point and the resulting control laws are analyzed in detail. In addition, we investigated the application to the Lorenz system [R47], and the results are found in the supplemental material.

4.3.1 *Harmonic Oscillator*

Consider a harmonic oscillator incorporated into the feedback control scheme discussed in Section 4.2.1. The dynamical system reads:

$$\ddot{x} = -\omega_0^2 x + u(x, \dot{x}), \tag{4.3}$$

with the particular system parameters defined in the left half of Table 4.2. Ideal sensors are assumed to measure position and velocity, $\vec{s} = (x, \dot{x})$; thus, the explicit statement of the sensor function is omitted in the argument list of u.

The control target is to drive the harmonic oscillator toward a steady state resting position. This may be formulated into a cost function using the root mean squared error (RMSE) of the trajectory x with reference to 0, that is,

$$\Gamma_1 := \text{RMSE}(x, 0) = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} (x(t_i) - 0)^2}.$$

Since this is a numerical experiment, the RMSE is formulated in a discrete form, with time steps $t_i = i\frac{T}{N}$ $(i = 0, \ldots, N-1)$, and an oscillation period $T = \frac{2\pi}{\omega_0}$. A time interval of twenty periods, as defined in Table 4.2, is large enough to get a meaningful measurement of Γ_1. The number of discrete time steps, n, is chosen such that an accuracy of 50 samples per period is achieved.

Further, as discussed in Fig. 4.2.2, the expression length is used as the second objective to bias GP learning towards smaller control laws:

$$\Gamma_2 := \text{length}(u),$$

which, corresponds to the number of nodes in the expression tree for u.

Table 4.2: Harmonic oscillator system setup.

Dynamic system		GP	
ω_0	$\exp(2)$	Cost functionals	$\text{RMSE}(x, 0)$
$x(t_0)$	$\ln(4)$		$\text{Length}(u)$
$\dot{x}(t_0)$	0	Argument set	$\{x, \dot{x}\}$
t_0, t_n	0, $20\frac{2\pi}{\omega_0}$	Constant set	$\{k\}$
n	1000	Seed	1730327932332863820

These two cost functions are listed as part of the GP setup in the right half of Table 4.2. Additionally, the argument set and constant set are specified. In this case, the argument set consists of symbols representing position and velocity, the two quantities measured by $\vec{s}$; the constant set consists of a single constant, k, that is used to perform constant optimization. For easier readability, the notation does not distinguish between primitive symbols and variable names, that is, x is used instead of `x`, and $\dot{x}$ instead of `x_dot`. The seed from Table 4.2 is used to initialize the pseudorandom number generator of the GP algorithm and leads to the particular solutions presented next. Other relevant parameters are stated in the general GP setup, as described in Fig. 4.2.2.

The Pareto solutions for this particular setup are presented in Table 4.3 in ascending order sorted by Γ_1 (Root Mean Squared Error (RMSE)). Not surprisingly, more complex expressions tend to provide better control (in terms of lower RMSE). One can also observe multiple mathematically equivalent solutions in the Pareto set (e. g. , the two expressions of length 6). Although equivalent, these expressions are distinct as far as the internal representation is concerned and the GP algorithm treats them as independent solutions.

Table 4.3: Control of the harmonic oscillator: Pareto-front solutions.

RMSE	Length	Expression	Constants
0.043973	10	$k \cdot (-k \cdot x + x - \dot{x})$	$k = 151.907120$
0.043976	8	$k \cdot (-k \cdot x - \dot{x})$	$k = 151.232316$
0.115862	7	$\exp(-k \cdot x - \dot{x})$	$k = 3509.921747$
0.123309	6	$k \cdot (-x - \dot{x})$	$k = 8.545559$
0.123309	6	$-k \cdot (x + \dot{x})$	$k = 8.545559$
0.123309	5	$k \cdot (x + \dot{x})$	$k = -8.545254$
0.127432	3	$k \cdot \dot{x}$	$k = -7.389051$
0.241743	2	$-\dot{x}$	
0.801177	1	k	$k = 25.229759$

Fig. 4.7 shows the trajectories of two particular solutions from Table 4.3 (first row and the third row from the bottom). Both look like underdamped cases of the damped harmonic oscillator system. We will analyze both solutions in more detail.

First consider $u(x, \dot{x}) = k(-kx + x - \dot{x})$. Inserting into the general Equation 4.3 for the controlled harmonic oscillator, one gets

$$\begin{aligned} \ddot{x} &= -\omega_0^2 x + k(-kx + x - \dot{x}) \\ &= -(\omega_0^2 + k^2 - k)x - k\dot{x} \\ &= -\tilde{\omega}_0^2 x - k\dot{x} \end{aligned} \tag{4.4}$$

with $\tilde{\omega}_0^2 := \omega_0^2 + k^2 - k$. This is indeed the differential equation for the damped harmonic oscillator. Since $\omega_0^2 > 1$, it follows that $\tilde{\omega}_0^2 > 0$ and the condition for the

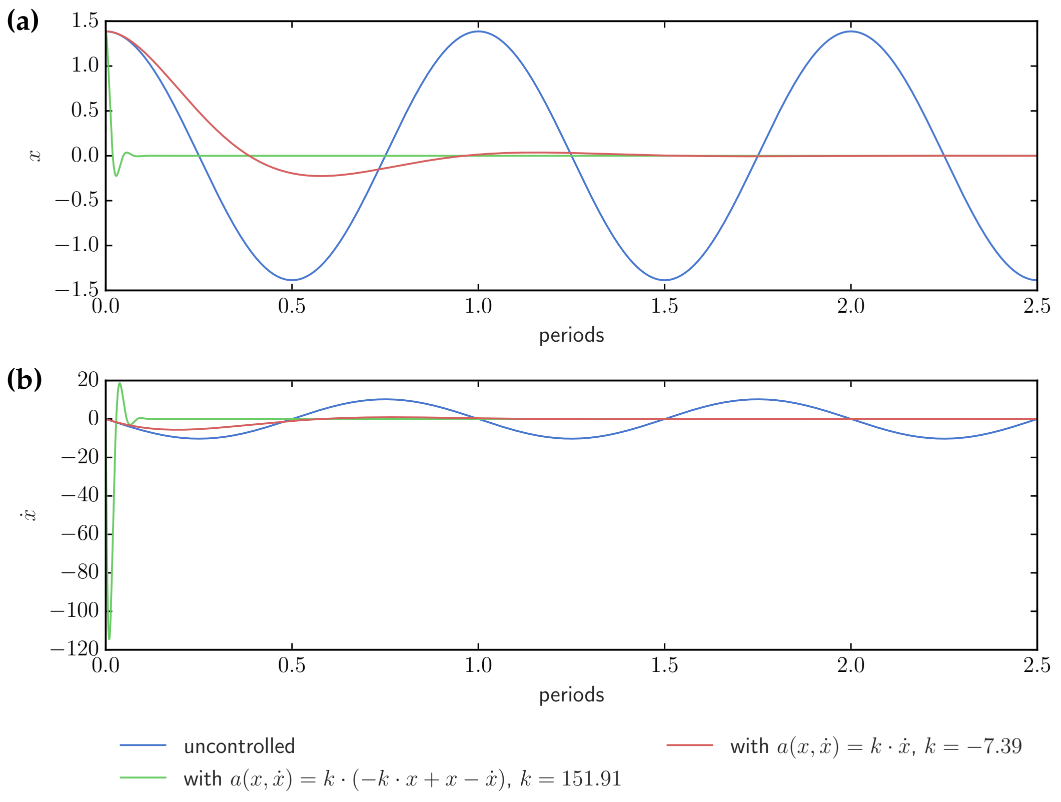


Figure 4.7: Control of the harmonic oscillator. The trajectories of two candidate solutions, chosen from the Pareto front that drive the harmonic oscillator to zero are shown. The best solution regarding Γ_1 are shown in a green color. A simple yet moderately good solution with respect to Γ_1 is shown in red. For reference the uncontrolled system is shown in blue. **a** Position, **b** speed of the oscillators.

underdamped case, $\frac{k^2}{4} < \tilde{\omega}_0^2$, is fulfilled for any $k \in \mathbb{R}$. Using the initial values from Table 4.2, we get the particular solution

$$x(t) = e^{-\frac{k}{2}t+2} \left(\cos(\omega t) + \frac{k \sin(\omega t)}{2\omega} \right), \tag{4.5}$$

where $\omega^2 := \tilde{\omega}_0^2 - \frac{k^2}{4}$.

The form of solution (4.5) demands an answer to the specific value found for k: One would expect large values of $k \gg 151.9$ (up to the numeric floating point limit), since one of the goals of optimization is to drive the harmonic oscillator to zero, and the particular solution (4.5) implies that $x(t) \to 0$ as $k \to \infty$ ($t \in \mathbb{R}$). Why the least squares algorithm finds a considerably smaller value can be explained by the choice of discretization made here. Fig. 4.8 illustrates how the optimal value k^* depends on the discretization of the finite time interval, more specifically the step size Δt. Since the result of numerical integration is restricted by the resolution of the time interval, starting at the initial position $x(0) = e^2$, the shortest time span possible for the trajectory to reach zero is Δt. Thus, setting an upper bound k^* when optimizing for the RMSE of the whole trajectory with respect to zero: values larger k^* would not improve the cost index any further.

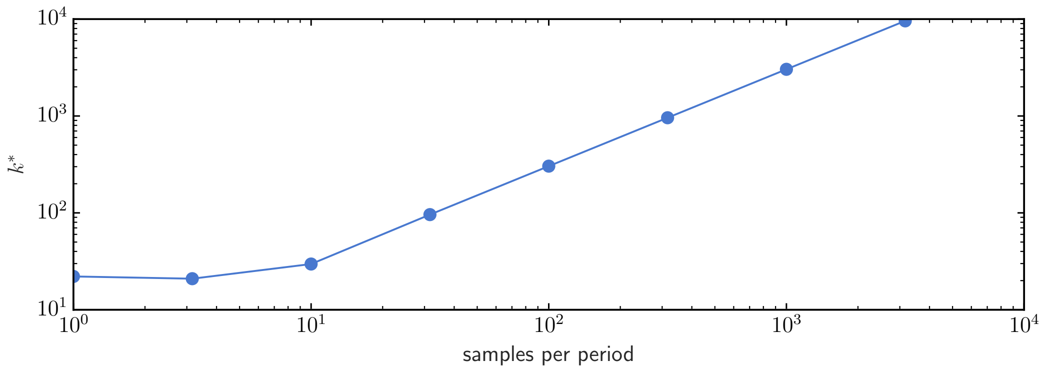


Figure 4.8: Control of the harmonic oscillator. Optimal parameter k^* for the control law $u(x, \dot{x}) = k \cdot (-k \cdot x + x - \dot{x})$ as determined by the least squares optimization using different sampling rates f_s. Starting at $f_s = 10$ the optimal value for k increases linearly with the sampling rate, $k^* \sim f_s \sim 1/\Delta t$.

As the second case, consider the solution $u(x, \dot{x}) = k \cdot \dot{x}$. Again inserting into (4.3)

$$\ddot{x} = -\omega_0^2 x + k\dot{x}, \tag{4.6}$$

one gets a damped harmonic oscillator system. The difference to the previous case (4.5), is that the coefficient of x does not depend on k. This allows for a wider range of solutions that cover all regimes of the damped harmonic oscillator (i.e., overdamped, underdamped, and diverging case). A numerical analysis of the RMSE with respect to k shows the presence of a single minimum in the underdamped regime; see Fig. 4.9. The result $k^* = -\omega_0$ from constant optimization corresponds almost exactly to the minimum position, as would be expected.

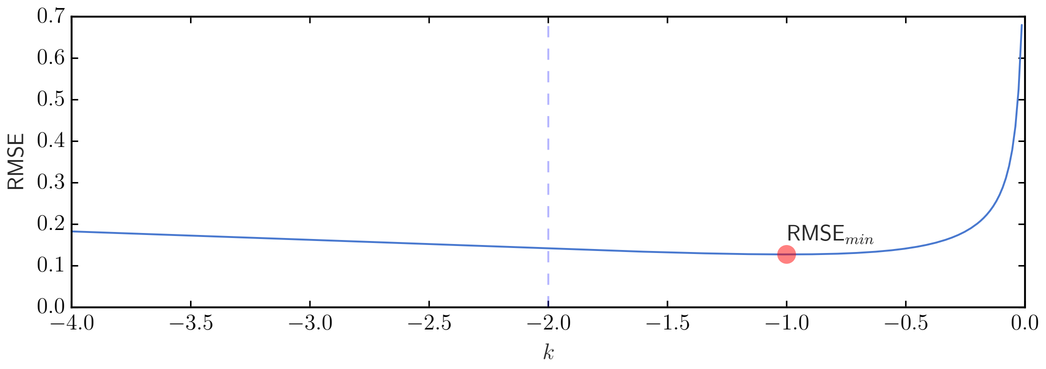


Figure 4.9: Control of the harmonic oscillator. Minimum in RMSE with respect to k for the control law $u(x, \dot{x}) = -k \cdot \dot{x}$. The minimum, at $k = -7.389 \approx -\omega_0$, lies in the underdamped regime. Solutions in the overdamped regime, left of the aperiodic borderline case (dashed line), result in strictly increasing RMSE as $k \to -\infty$. So do the diverging solutions for $k > 0$, as $k \to \infty$ (not shown).

4.4 FROM COUPLED OSCILLATORS TO NETWORKS

In this section, we extend the above study and demonstrate the application of GP-based control to networks of oscillators. As discussed in the introduction, such networks are used to model highly nonlinear complex systems, including the human brain. Given the latter as an application, the target structure is a hierarchical network. Nevertheless we want to systematically investigate the results of our method starting with a well-understood situation, namely two coupled oscillators. To step toward a network structure, we extend this first to a one-dimensional ring, or chain of coupled oscillators with periodic boundary conditions. Then we consider the two-dimensional analogon, the torus. Eventually, we study a hierarchical network which has been proposed as a simplified model for the human brain [R48]. The two coupled oscillators and the network results are discussed below; 1D and 2D periodic structures are discussed in supplemental material, since there is no essential new information in the results. Nevertheless one recognizes that results are consistent.

The aim, for all systems under consideration, is to control the synchronization behavior of the coupled oscillators. This can be done in two ways: starting from a synchronization regime and forcing the system into de-synchronization or vice versa, i.e., starting from a de-synchronized regime and forcing the system into synchronization. Both control goals are evaluated for the systems investigated.

The synchronization of dynamical systems is a well-known phenomenon exhibited by diverse ensembles of oscillators and oscillatory media [R6]. Here we will focus on a simple, but popular representative, the van der Pol oscillator, also used as a simple model for neurons. The van der Pol oscillator shows a nonlinear damping behavior governed by the following second-order differential equation:

$$\ddot{x} = -\omega^2 x + \alpha \dot{x} \left(1 - \beta x^2\right) =: f_{\mathrm{vdP}}(x, \dot{x}), \tag{4.7}$$

where x is the dynamical variable and ω, α, $\beta > 0$ are model parameters. The parameter ω is the characteristic frequency of the self-sustained oscillations, that is, the frequency at which the system tends to oscillate in the absence of any driving or damping force. The parameter α controls the non-linearity of the system. When $\alpha = 0$, Eq. (4.7) becomes the harmonic oscillator. The damping parameter β controls the dilation of the trajectory in the phase space. See Fig. 4.10.

A wide variety of coupling mechanisms exist. For hierarchical networks the investigation can be restricted to linearly coupled van der Pol oscillators in x and $\dot{x}$, which can be described by a global coupling constant and two coupling matrices. An uncontrolled system of N coupled van der Pol oscillators can be stated as follows:

$$\ddot{x}_i = f_{\mathrm{vdP}}(x_i, \dot{x}_i) + c_1 \sum_{j=0}^{N-1} \kappa_{ij} x_j + c_2 \sum_{j=0}^{N-1} \varepsilon_{ij} \dot{x}_j \qquad (i = 0, \ldots, N-1) \tag{4.8}$$

with initial conditions

$$x_i(t_0) = x_{i,0}, \qquad \dot{x}_i(t_0) = \dot{x}_{i,0},$$

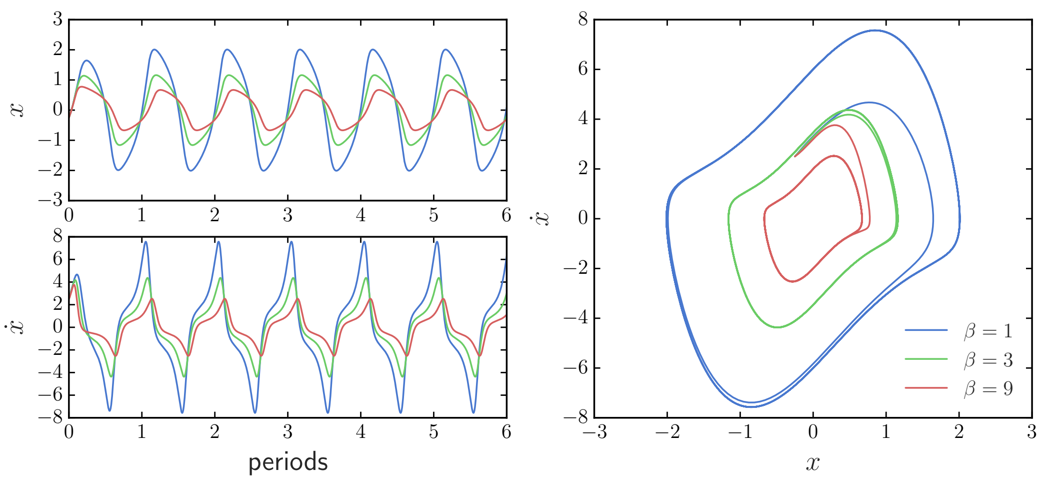


Figure 4.10: Single van der Pol oscillator (with parameters $\omega = e^2$, $\alpha = 3$, and initial conditions $x(0) = -0.25$, $\dot{x}(0) = 2.5$.)

where $c_{1,2}$ are the global coupling constants and (κ_{ij}) and (ε_{ij}) are the respective coupling matrices. This allows for several types of coupling such as direct, diffusive, and global coupling, or any other kind of network-like coupling. In the following experiments, we will use diffusive coupling in $\dot{x}_i$ for the topologies mentioned above. For GP, we use the same setup described above (Fig. 4.2.2).

4.4.1 *Two Coupled Oscillators*

The simplest system showing synchronization is a system of two dissipatively coupled van der Pol oscillators [R6]:

$$\begin{aligned} \ddot{x}_0 &= -\omega_0^2 x_0 + \alpha \dot{x}_0 \left(1 - \beta x_0^2\right) + c\left(\dot{x}_1 - \dot{x}_0\right), \\ \ddot{x}_1 &= -\omega_1^2 x_1 + \alpha \dot{x}_1 \left(1 - \beta x_1^2\right) + c\left(\dot{x}_0 - \dot{x}_1\right). \end{aligned} \tag{4.9}$$

The coupling is restricted to $\dot{x}_i$, in which case the coupling constants from (4.8) are set to $c_1 = 0$, $c_2 = c$, and the remaining coupling matrix reads as follows: $(\varepsilon_{ij}) = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$.

For the uncoupled oscillators, there are some parameter combinations (α, β) for which there exist stable limit cycles with characteristic frequencies $\omega_{0,1}$. If the two oscillators are coupled by a given coupling constant $c \neq 0$, as in (4.9), a range of frequencies with $\omega_0 \neq \omega_1$ emerge, where both oscillators effectively oscillate in a common mode. This range of frequencies is called the synchronization region. With the variation in the coupling constant, this region changes in width.

To illustrate this phenomenon, consider the particular parameter set $\alpha = 0.1$, $\beta = 1$, with fixed $\omega_0 = 1.386$ and varying ω_1 in the range $[\omega_0 - 0.06, \omega_0 + 0.06]$. By plotting the observed frequency difference[1] $\Delta\Omega$, exhibited by the two oscillators, against the difference in their characteristic frequencies, $\Delta\omega := \omega_1 - \omega_0$, we can

1 The actual frequency Ω of an oscillator can be determined numerically by either taking the Fourier transform of the trajectory $x(t)$, or by counting the zero-crossings of $x(t) - \langle x(t) \rangle$.

visualize the synchronization behavior of the system for a given coupling constant. See Fig. 4.11. Regions of synchronization show up as horizontal segments at $\Delta\Omega = 0$ (also, note the symmetry about $\Delta\omega = 0$). If this is done for several values of c in the range $[0, 0.4]$, we can trace out the regions of synchronization. The result is a typical V-shaped plateau, the Arnold tongue.

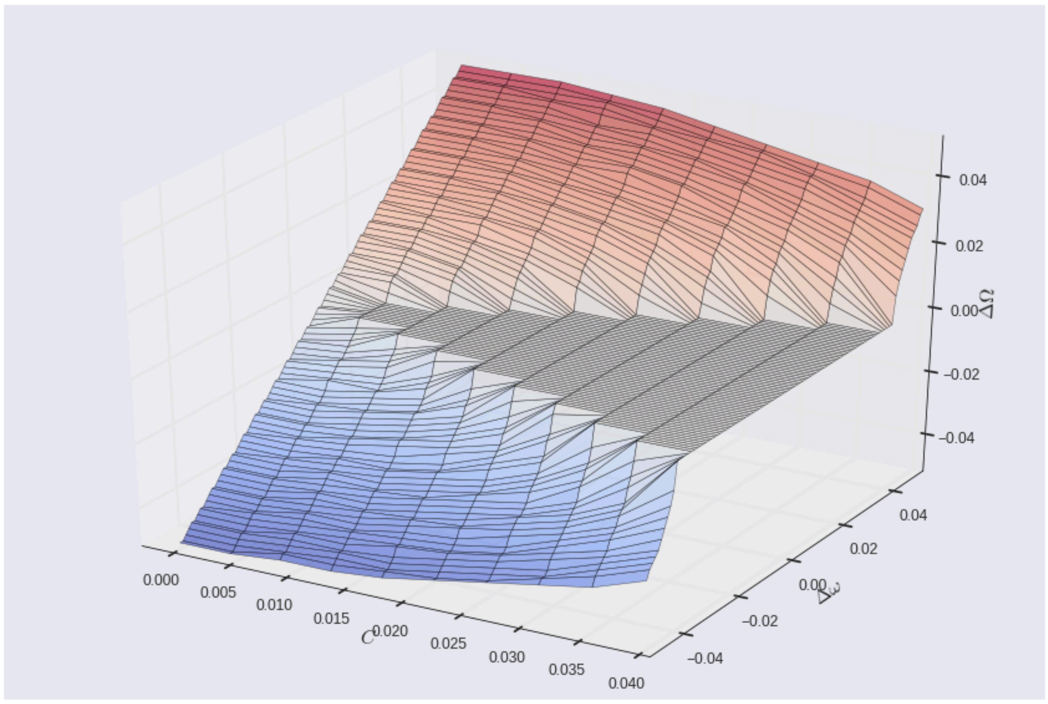


Figure 4.11: Synchronization plot of two coupled van der Pol oscillators with varying coupling strength c. The horizontal V-shaped plateau is referred to as the Arnold tongue; it represents regions of synchronization (The parameter set and initial conditions used are stated in the left part of Table 4.4).

Fig. 4.11 shows the choice of appropriate parameters ω_1 and c to set up the system in different regimes for the purpose of control. The same approach is taken for all the experiments presented in this section.

Similar to the example systems in Section 4.3 we add the control function u to the equations (4.9) of the uncontrolled system, yielding the following formulation:

$$\begin{aligned} \ddot{x}_0 &= -\omega_0^2 x_0 + \alpha \dot{x}_0 \left(1 - \beta x_0^2\right) + c\left(\dot{x}_1 - \dot{x}_0\right) + u(\dot{\vec{x}}), \\ \ddot{x}_1 &= -\omega_1^2 x_1 + \alpha \dot{x}_1 \left(1 - \beta x_1^2\right) + c\left(\dot{x}_0 - \dot{x}_1\right) + u(\dot{\vec{x}}). \end{aligned} \tag{4.10}$$

Here, u is added as a global actuator term with equal influence on both oscillators; u may depend on $\dot{x}_0$ and $\dot{x}_1$, summarized in vector notation as $\dot{\vec{x}} = (\dot{x}_0, \dot{x}_1)$.

Forced Synchronization

The system setup for forced synchronization of the two coupled van der Pol oscillators is presented in Table 4.4. The parameters ω_1 and c are chosen according to Fig. 4.11, such that the uncontrolled system follows a de-synchronization regime at a distance, $\Delta\omega$, approximately half the plateau from the closest synchronization point. The initial conditions are the same for both oscillators. The stopping criteria are chosen heuristically; in particular, the runtime was chosen from preliminary runs such that a typical control law can be found within that number of iterations.

The degree of de-synchronization is encompassed by the following cost functional:

$$\Gamma_1 := |\Omega_0 - \Omega_1|. \tag{4.11}$$

It measures the difference in observed frequencies exhibited by the two oscillators, with smaller differences reducing the cost on this objective.

As stated in the previous section, the actual frequencies Ω_0 and Ω_1 are numerically determined by counting zero crossings of the trajectory $x - \langle x \rangle$. This requires a careful choice of the time range $[t_0, t_n]$ of the observations, since the number of periods N_P fitting into this interval determines an upper bound in absolute accuracy ($\sim \frac{1}{2N_P}$) of measuring Ω_0, Ω_1. Here, $N_P = 2000$ is chosen to yield an absolute accuracy well below 10^{-3} in the frequency range of interest.

Table 4.4: Two coupled oscillators: system setup for forced synchronization.

Dynamic system		GP	
ω_0	$\ln(4)$	Cost functionals	$\lvert\Omega_0 - \Omega_1\rvert$
ω_1	$\ln(4) + 0.04$		$\text{Length}(u)$
α, β, c	0.1, 1, 0.022	Argument set	$\{\dot{x}_0, \dot{x}_1\}$
$\vec{x}(t_0)$	$(1,1)$	Constant set	$\{k\}$
$\dot{\vec{x}}(t_0)$	$(0,0)$	Seed	3464542173339676227
t_0, t_n	$0,\ 2000\frac{2\pi}{\omega_0}$		
n	40000		

Results from the GP run are presented in Table 4.5. The algorithm stopped after one generation, providing six simple results optimally satisfying Γ_1. Since the equations (4.10) are symmetric in x_0 and x_1, unsurprisingly so are the resulting control laws.

To demonstrate the control effect Fig. 4.12 shows the Kuramoto order parameter, r, representing phase coherence, plotted over time [R49, R50]. The order parameter is defined by

$$r = \left| \frac{1}{N} \sum_{j=0}^{N-1} e^{i\varphi_j} \right|,$$

with φ_j the continuous phase of the jth oscillator. It is computed from the analytical signal of the trajectory x using the Hilbert transform, cf.[R51]. The controlled system completely synchronizes ($r \approx 1$) after a short initial period of de-synchronization, while the uncontrolled system exhibits an oscillating graph.

One recognizes that the algorithm favors the least complex solution, using an asymmetric term, either damping in x_1 or x_0. We can analyze qualitatively these so-

Table 4.5: Two coupled oscillators: Pareto-front solutions for forced synchronization.

$\lvert\Omega_0 - \Omega_1\rvert$	Length	Expression
0.0	2	$\cos(\dot{x}_1)$
0.0	2	$\cos(\dot{x}_0)$
0.0	2	$-\dot{x}_0$
0.0	2	$\sin(\dot{x}_1)$
0.0	2	$-\dot{x}_1$
0.0	2	$\sin(\dot{x}_0)$

lutions, $u(\dot{\vec{x}}) = -\dot{x}_0$ and $u(\dot{\vec{x}}) = -\dot{x}_1$. Let us take arbitrarily the x_0 term: Plugging in $u(\dot{\vec{x}}) = -\dot{x}_0$ into the first oscillator equation from (4.10) we obtain

$$\begin{aligned}\ddot{x}_0 &= -\omega_0^2 x_0 + \alpha\dot{x}_0\left(1 - \beta x_0^2\right) + c\left(\dot{x}_1 - \dot{x}_0\right) - \dot{x}_0\\ &= -\omega_0^2 x_0 + (\alpha - c - 1)\dot{x}_0 - \alpha\beta\dot{x}_0 x_0^2 + c\dot{x}_1\\ &= -\omega_0^2 x_0 + \tilde{\alpha}_1\dot{x}_0 - \alpha\beta\dot{x}_0 x_0^2 + c\dot{x}_1,\end{aligned}$$

with $\tilde{\alpha}_1 = \alpha - c - 1$. Doing the same for the second oscillator equation

$$\begin{aligned}\ddot{x}_1 &= -\omega_1^2 x_1 + \alpha\dot{x}_1\left(1 - \beta x_1^2\right) + c\left(\dot{x}_0 - \dot{x}_1\right) - \dot{x}_0\\ &= -\omega_1^2 x_1 + (c - 1)\dot{x}_0 - \alpha\beta\dot{x}_1 x_1^2 + (\alpha - c)\dot{x}_1\\ &= -\omega_1^2 x_1 + \tilde{\alpha}_2\dot{x}_0 - \alpha\beta\dot{x}_1 x_1^2 + \tilde{c}\dot{x}_1,\end{aligned}$$

with $\tilde{\alpha}_2 = c - 1$ and $\tilde{c} = \alpha - c$, one gets a similar solution in terms of the dominating driving component $\dot{x}_0$. Since $\tilde{\alpha}_1 \approx \tilde{\alpha}_2$ and $\tilde{\alpha}_1 \gg \tilde{c}, c$, the oscillators behave almost identically and, thus, are synchronized. Analogous observations apply when analyzing the second control law, $u(\dot{\vec{x}}) = -\dot{x}_1$.

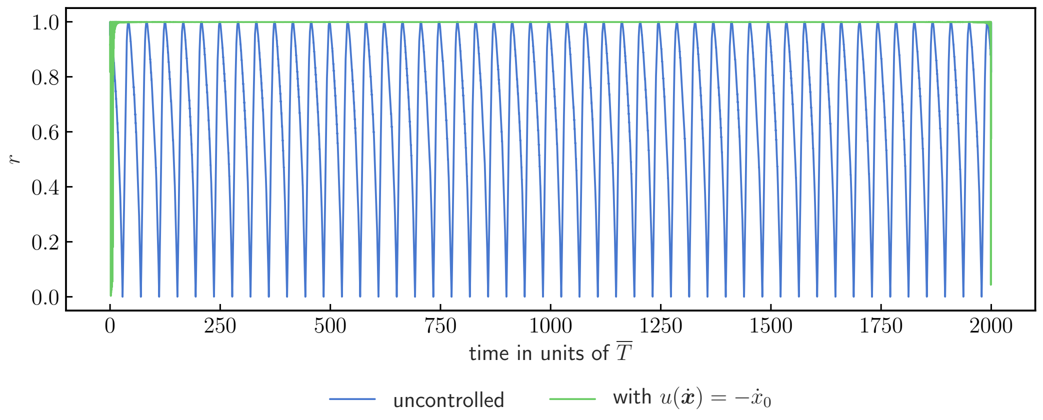


Figure 4.12: Two coupled oscillators: Kuramoto order parameter, r, for forced synchronization. Green: the controlled and blue: the uncontrolled system.

Forced De-Synchronization

The system setup for forced de-synchronization is given in Table 4.6. The parameters ω_1 and c are, again, chosen according to Fig. 4.11, this time, such that the uncontrolled system follows a synchronization regime well inside the plateau. The measure for the degree of synchronization is reciprocal to the previous case

$$\Gamma_1 := \exp(-|\Omega_0 - \Omega_1|). \tag{4.12}$$

All the remaining parameters are the same as for forced synchronization (Table 4.4).

Table 4.6: Two coupled oscillators: System setup for forced de-synchronization.

Dynamic system		GP	
ω_0	$\ln(4)$	Cost functionals	$\exp(-\lvert\Omega_0 - \Omega_1\rvert)$
ω_1	$\ln(4) + 0.015$		$\mathrm{Length}(u)$
α, β, c	0.1, 1, 0.022	Argument set	$\{\dot{x}_0, \dot{x}_1\}$
$\vec{x}(t_0)$	$(1, 1)$	Constant set	$\{k\}$
$\dot{\vec{x}}(t_0)$	$(0, 0)$	Seed	2590675513212712687
t_0, t_n	$0,\ 2000\frac{2\pi}{\omega_0}$		
n	40000		

Results from the GP run are shown in Table 4.7. Two aspects of the results indicate that de-synchronizing the pair of oscillators is a more demanding task: First, the GP algorithm comes up with increasingly long expressions to achieve improvements in Γ_1; second, constant optimization seems to fail in all cases where a constant is present. (This is expressed by a value $k = 1$, which corresponds to the initial guess of the optimization procedure.) Still, the oscillating Kuramoto parameter, r, of the controlled system in Fig. 4.13 shows that the best solution with respect to Γ_1 performs well in de-synchronizing the oscillators.

The control law $u(\dot{\vec{x}}) = -\dot{x}_0 \cdot \exp(\exp(k) + \cos(k)) = -\tilde{k}\dot{x}_0$, with $\tilde{k} \approx 26$, is almost the same as the solution analyzed in the previous subsection, but with a different coefficient. This is at first sight counterintuitive, but can be explained roughly by the non-uniqueness we provoke with our cost function: To force synchronization, we require only that the phase difference is small (close to zero). This is achieved by the added damping term. The very strong damping brings the two oscillators basically to zero so fast that the mutual coupling does not play a role and the phase difference is free. To bring the oscillators from de-synchronization to synchronization is achieved by a different mechanism; the damping is moderate such that excess energy is dissipated and the oscillators are in the right regime to synchronize. The detailed analysis of the dynamics is subject of ongoing work, where we analyze the bifurcations occurring using AUTO. Preliminary results affirm that the interpretation given here is correct.

One would expect the GP algorithm to directly generate the simpler — thus better suitable— solution $u(\dot{\vec{x}}) = -k\dot{x}_0$, with $k = 26$. One possible explanation why this is not immediately found might lie in the constant optimization algorithm: On failure, the least squares algorithm returns the result of the last internal iteration. This return value might be an entirely inadequate value for k, which, in turn, could lead to an exploding cost index Γ_1 when integrating the dynamic system (4.10), hence disqualifying the corresponding solution.

Table 4.7: Two coupled oscillators: Pareto-front solutions for forced de-synchronization.

$\exp(-\lvert\Omega_0 - \Omega_1\rvert)$	Length	Expression	Constant
0.248	9	$-\dot{x}_0 \cdot \exp(\exp(k) + \cos(k))$	$k = 1$
0.258	7	$\cos(\exp(\dot{x}_1 + \cos(\cos(\dot{x}_0))))$	
0.875	4	$\cos(\exp(\exp(\dot{x}_0)))$	
0.912	3	$\sin(\exp(\dot{x}_0))$	
0.999	2	$\exp(k)$	$k = 1$
1.000	1	$\dot{x}_1$	
1.000	1	$\dot{x}_0$	
1.000	1	k	$k = 1$
0.247672	13	$-\dot{x}_0 \cdot \exp(-\dot{x}_0) \cdot \exp(\exp(k)) - \exp(\dot{x}_0)$	$k = 1.00$
0.386623	12	$-\exp(\dot{x}_0) - \cos(\exp(\exp(k)) \cdot \exp(\sin(\dot{x}_0)))$	$k = 1.00$
0.398873	11	$\sin(\exp(\exp(\dot{x}_0) + \cos(k)) \cdot \exp(\sin(\dot{x}_0)))$	$k = 1.00$
0.606677	8	$\sin(\exp(\exp(k)) \cdot \exp(\sin(\dot{x}_0)))$	$k = 1.00$
0.648420	7	$\sin(\exp(\exp(\dot{x}_0) + \cos(k)))$	$k = 1.00$
0.806083	6	$-\exp(\dot{x}_0) - \cos(k)$	$k = 1.00$
0.911933	3	$\sin(\exp(\dot{x}_0))$	$k = 1.00$
0.998615	2	$\exp(k)$	$k = 1.00$
1.000000	1	$\dot{x}_1$	$k = 1.00$
1.000000	1	$\dot{x}_0$	$k = 1.00$
1.000000	1	k	$k = 1.00$

4.4.2 *Hierarchical Network*

In a last step the dynamic system is extended to a set of van der Pol oscillators connected in a scale-free network topology. A scale-free network is a hierarchical network whose degree distribution follows a power law, at least asymptotically. There exists a large variety of possible models for creating networks which are able to reproduce the unique properties of the scale-free topology. One simple model, resorted to here, is the Dorogovtsev–Goltsev–Mendes model. It is used to produce a network of $N = 123$ nodes as depicted in Fig. 4.14.

The van der Pol oscillators are indexed in descending order by their corresponding node degree. For example, the three yellow nodes of degree 32 (highest) in Fig. 4.14 are labeled $i = 0, 1, 2$, the light orange nodes of the next lowest degree 16, $i = 3, 4, 5$, and so on. The particular order of nodes of the same degree is not important due to the symmetry of the network.

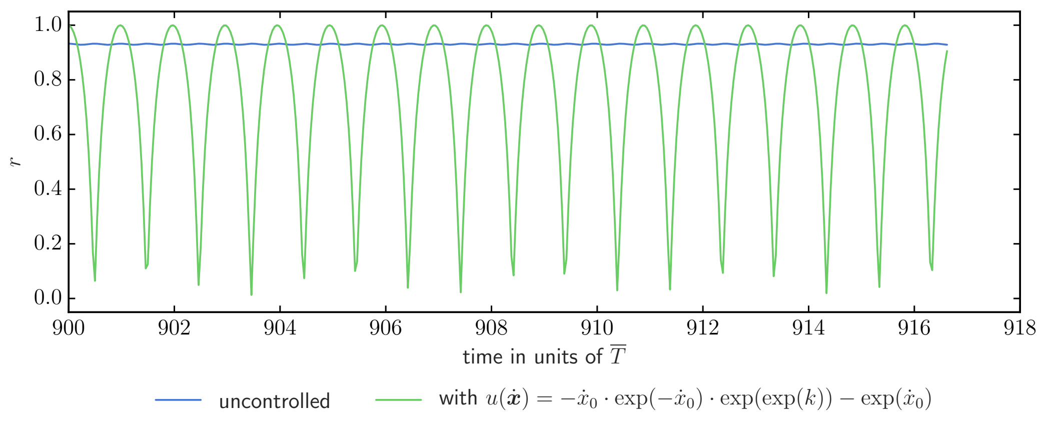


Figure 4.13: Two coupled oscillators: Kuramoto order parameter, r, for forced de-synchronization. Green: the controlled and blue: the uncontrolled system. The horizontal axis is scaled to a limited time window in order to make the oscillations visible.

"Sensors" are placed on the oscillators labeled $i = 0, \ldots, 11$ measuring $\dot{x}_i$. This incorporates all nodes with a node degree of 32 or 16, and five nodes with a node degree of 8. Hence, the control function u can potentially make use of measurements from nodes at central points of the topology. Control, on the other hand, is exerted indiscriminately on all nodes of the system

Forced Synchronization

Table 4.8 shows the setup. As in the related cases before, the particular parameter set chosen puts the uncontrolled system in a de-synchronization regime. Sensors are placed on the oscillators labeled $i = 0, \ldots, 11$ measuring $\dot{x}_i$. This incorporates all nodes with a node degree of 32 or 16, and five nodes with a node degree of 8. Hence, the control function u can potentially make use of measurements from nodes at central points of the topology. Control, on the other hand, is excerted indiscriminately on all nodes of the system

$$\ddot{x}_i = f_{\mathrm{vdP}}(x_i) + c \sum_{j=1}^{N} \varepsilon_{ij} \dot{x}_j + u(x_0, \ldots, x_{11}) \qquad (i = 0, \ldots, N-1),$$

with $N = 123$. Constant optimization is performed on a single constant k.

The GP run stopped after one generation with a Pareto front consisting of a single optimal result. See Table 4.9. The control law found, is again, sinusoidal in nature. It uses the input from node $i = 7$, which is of node degree eight, i.e., on an intermediate level in the topology. Fig. 4.15 shows that the highly distorted phases from the uncontrolled system can be partially aligned. Frequencies are approximately matched and amplitudes are amplified by a factor $\times 1.5$ with respect to the uncontrolled system. A plot of the Kuramoto order parameter r in Fig. 4.16 indicates that there is a small variation among the phases in the controlled system; hence, a perfect phase lock is not achieved.

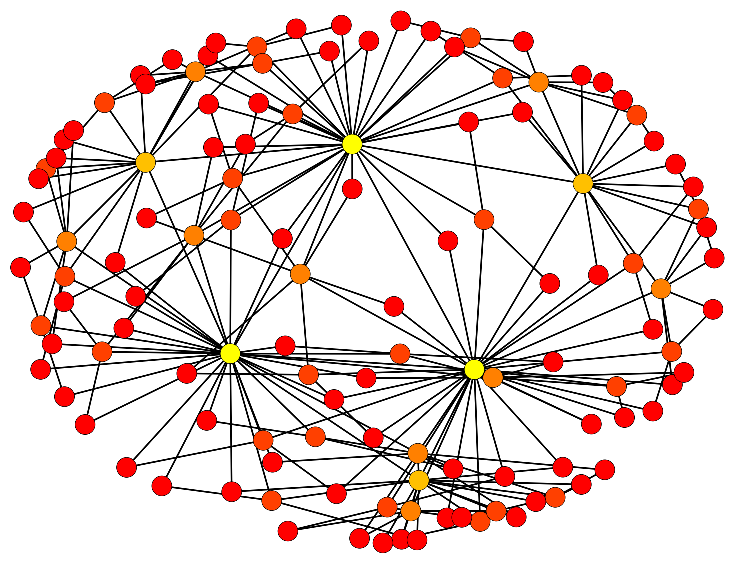


Figure 4.14: Hierarchical network: Dorogovtsev–Goltsev–Mendes topology of generation five. Starting out from two connected nodes at generation 0, one new node is added in between every existing pair of nodes per generation. Hence, the node degree at generation $n > 0$ ranges from $2^1, \dots, 2^n$. The degree distribution, that is, the fraction $P(k)$ of nodes in the network having k connections to other nodes goes for large values of k as $P(k) \sim k^{-2}$ (Nodes are color-coded by node degree: yellow: 32, light orange: 16, orange: 8, dark orange: 4, red: 2).

Table 4.8: Oscillators in a hierarchical network: system setup for forced synchronization.

Dynamic system		GP	
α, β, c	$0.1,\ 1,\ 5.6 \cdot 10^{-2}$	Cost functionals	$\mathrm{std}(\vec{\Omega})$
$\overline{\omega}$, $\Delta\omega$	$\ln(4),\ 8 \cdot 10^{-2}$		$\mathrm{Length}(u)$
ω_i	$\mathrm{linspace}(\overline{\omega} - \Delta\omega, \overline{\omega} + \Delta\omega, 123)$	Argument set	$\{\dot{x}_i\}_{i=0,\dots,11}$
$\vec{x}_i(t_0)$	$1\ (i = 0, \dots, 122)$	Constant set	$\{k\}$
$\dot{\vec{x}}_i(t_0)$	$0\ (i = 0, \dots, 122)$	Seed	5925327490976859669
t_0, t_n	$0,\ 2000\frac{2\pi}{\overline{\omega}}$		
n	40000		

Forced De-Synchronization

For the case of forced de-synchronization, the system parameters are set as in Table 4.10. The network structure stays as in the previous section.

The previous sections showed increasingly complex control from two oscillators, a ring of oscillators, to a torus. One might expect an even more complex control for

Table 4.9: Oscillators in a hierarchical network: Pareto-front solutions for forced synchronization.

std($\vec{\Omega}$)	length	expression
0.0	2	$\sin(\dot{x}_7)$

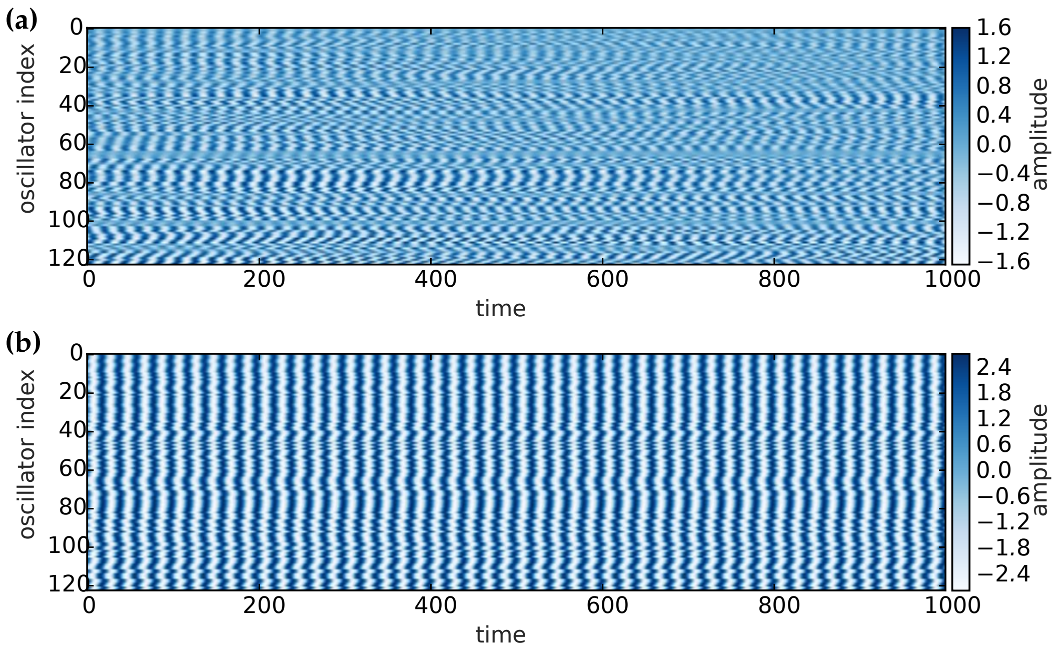


Figure 4.15: Oscillators in a hierarchical network. **a** uncontrolled system; **b** Pareto-front solution, $u(\dot{\vec{x}}) = \sin(\dot{x}_7)$, for forced synchronization.

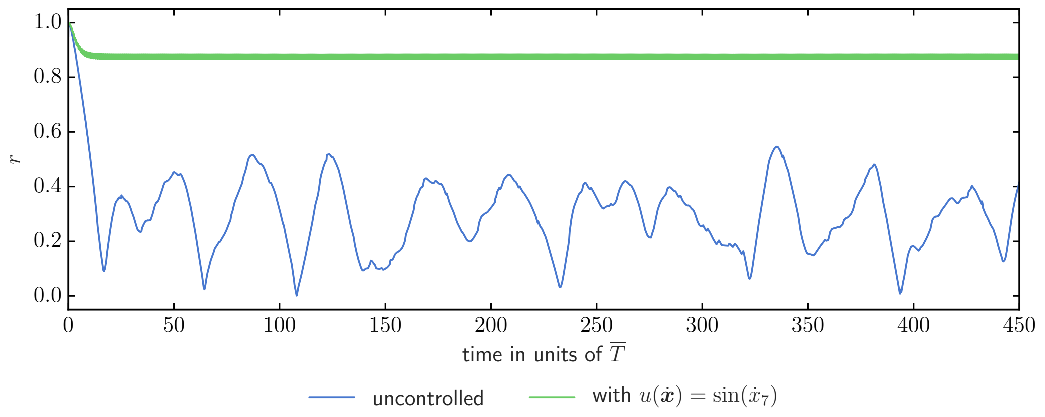


Figure 4.16: Oscillators in a hierarchical network. Kuramoto order parameter r for the uncontrolled system (blue) and the system controlled by the best solution, with respect to Γ_1 (green).

a hierarchical network. Indeed, in Table 4.11 the best control laws are complicated combinations of sine, multiplication and exponential terms. As explained above, the lowest indices indicate highest node degree. We observe that our algorithm puts control on nodes with high degree - the hubs. This is perfectly logical: If the

Table 4.10: Oscillators in a hierarchical network: system setup for forced de-synchronization.

Dynamic system		GP	
α, β, c	0.1, 1, $5.6 \cdot 10^{-2}$	Cost functionals	$\exp(\mathrm{std}(\vec{\Omega}))$
$\overline{\omega}$, $\Delta\omega$	$\ln(4)$, $2 \cdot 10^{-2}$		$\mathrm{Length}(u)$
ω_i	$\mathrm{linspace}(\overline{\omega} - \Delta\omega, \overline{\omega} + \Delta\omega, 123)$	Argument set	$\{\dot{x}_i\}_{i=0,\ldots,11}$
$\vec{x}_i(t_0)$	1 $(i = 0, \ldots, 122)$	Constant set	$\{k\}$
$\dot{\vec{x}}_i(t_0)$	0 $(i = 0, \ldots, 122)$	Seed	8797055239111497159
t_0, t_n	0, $2000\frac{2\pi}{\overline{\omega}}$		
n	40000		

hubs are de-synchronized, the whole system is desynchronized. It may still be that the hubs and their connected subnet are synchronized. This is not contained in our objective, and again, we find that our machine learns exactly the way it is told (Figs 4.16, 4.15). If we want to control global de-synchronization in each single oscillator, we need to design the cost function with more care!

Table 4.11: Oscillators in a hierarchical network: best solutions for forced de-synchronization.

Synchronicity	Length	Expression	Constant
0.722	17	$-(-\dot{x}_3 + (-\dot{x}_8)) + (\dot{x}_{11} + \dot{x}_8 + \sin(\dot{x}_9)) \cdot (-\exp(\dot{x}_0))$	
0.727	16	$-(-\dot{x}_3 + (-\dot{x}_8)) + (\dot{x}_{11} + \dot{x}_5 + \dot{x}_8) \cdot (-\exp(\dot{x}_{11}))$	
0.749	13	$\dot{x}_{11} + \dot{x}_8 + (\dot{x}_{11} + \dot{x}_8 + \dot{x}_9) \cdot (-\exp(\dot{x}_6))$	
0.886	4	$-\exp(\sin(\dot{x}_0))$	
0.886	3	$-\exp(\dot{x}_0)$	
0.997	2	$-\dot{x}_8$	
1.000	1	k	$k = 1$
1.000	1	$\dot{x}_4$	

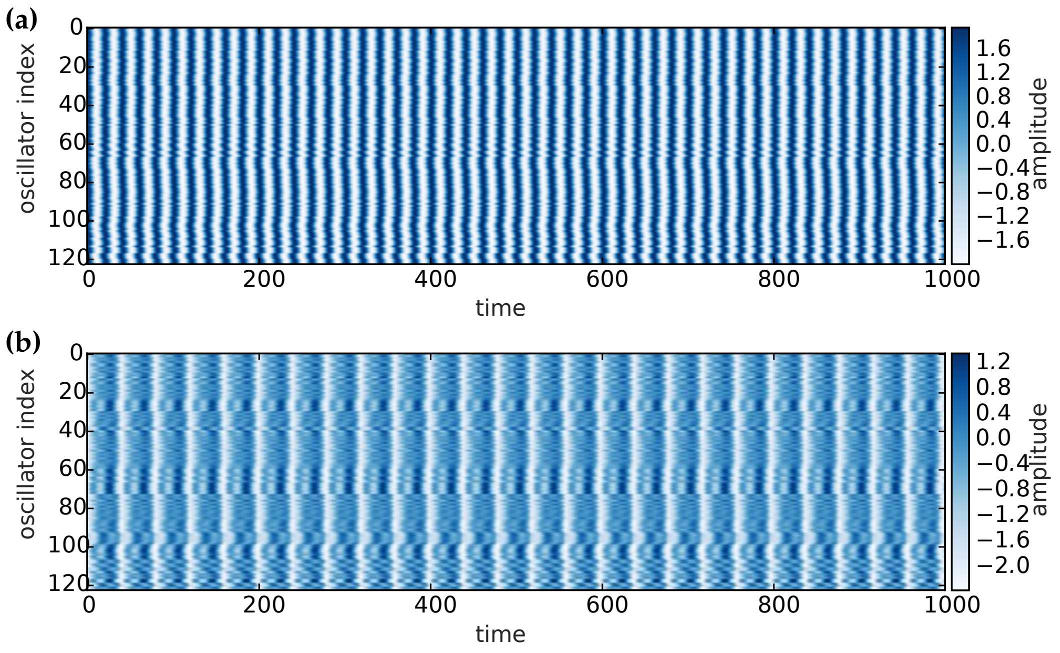


Figure 4.17: Oscillators in a hierarchical network. **a:** uncontrolled system; **b:** Pareto-front solution, $u(\vec{x}) = \dot{x}_3 + \dot{x}_8 - (\dot{x}_{11} + \dot{x}_8 + \sin(\dot{x}_9)) \cdot \exp(\dot{x}_0)$, for forced desynchronization.

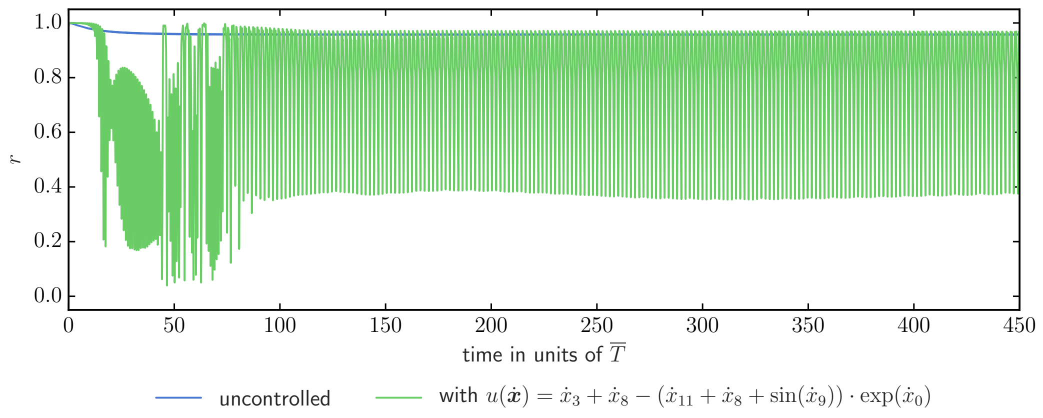


Figure 4.18: Oscillators in a hierarchical network. Kuramoto order parameter r for the uncontrolled system (blue) and the system controlled by the best solution, with respect to Γ_1 (green).

4.5 CONCLUSIONS AND FUTURE WORK

Our main question, in this work, concerns the control of a synchronization in systems of coupled oscillators. We presented a computational intelligence-based framework for inferring optimal control laws to achieve this goal. A multi-objective genetic programming algorithm with regression-based constant estimation is used to learn the control laws dynamically.

We first tested our method on a well-known control problem in dynamical systems: Drive a damped harmonic oscillator to a limit cycle and a stable one to a fixed point. We then applied our control approach to dynamical systems composed of networks of coupled oscillators, starting from a system of two coupled van der Pol oscillators up to a hierarchical network consisting of a few hundred oscillators. In comparison to other methods like generalized linear regression methods, GP is relatively complex to use for a beginner, however the effort pays off if general solutions are needed. Due to the evolutionary nature of the method, it is not guaranteed that the global optimum is found, which in our case was not a problem, but may well be in other situations.

As a result we find terms of different complexity leading to different levels of synchronization control, where synchronization is measured using the Kuramoto parameter. The results clearly demonstrate the ability of GP-based control to bring a desynchronized system to a synchronization state and vice versa. For forced synchronization, in any setup we find simple control laws, suggesting a single oscillator taking over the control and governing the overall dynamics. For forced desynchronization, laws of increasing complexity are found, where the complexity increased with that of the system complexity.

The results in all setups highlight the importance of designing the objective functions appropriately. In the current setup, we simply relied on learning the control laws by minimizing the error with the desired output. We did not specify any symmetry or energy function to be minimized, nor did we restrict the number of oscillators to be controlled. These details appear to be important for using our methods in a real-world application. The difficulty in the very general approach is recognized by the asymmetry of the control laws, which is a disadvantage in our opinion. One thus has to carefully analyze the objectives and may upgrade them step by step if unwanted solutions occur.

Further work will extend current methods in several ways: a subsequent automatic stability analysis would be performed for the numerical experiments. This way one can immediately distinguish stable and useful control dynamics from unstable ones. Second, we aim to look into the design of better objective functions, taking into consideration prior domain knowledge, e. g. in the form of additive symmetry terms. Finally, we plan to integrate methods in our framework that would search for the optimal sensor (for measurement) and pressure (for actuation) points in the network for a better control.

REFERENCES

[R1] N. Gautier, J.-L. Aider, T. Duriez, B. R. Noack, M. Segond, and M. Abel. "Closed-loop separation control using machine~learning." In: *J. Fluid Mech.* 770 (Apr. 2015), pp. 442–457. DOI: 10.1017/jfm.2015.95.

[R2] Edward Ott, Celso Grebogi, and James A. Yorke. "Controlling chaos." In: *Phys. Rev. Lett.* 64.11 (Mar. 1990), pp. 1196–1199. DOI: 10.1103/physrevlett.64.1196.

[R3] *Chaos Control*. Springer Berlin Heidelberg, 2003. DOI: 10.1007/b79666.

[R4] H. Haken. *Brain Dynamics: Synchronization and Activity Patterns in Pulse-Coupled Neural Nets with Delays and Noise*. Springer Series in Synergetics. Springer Berlin Heidelberg, 2006. URL: https://books.google.de/books?id=8elDAAAAQBAJ.

[R5] Jason M. Schwalb and Clement Hamani. "The history and future of deep brain stimulation." In: *Neurotherapeutics* 5.1 (Jan. 2008), pp. 3–13. DOI: 10.1016/j.nurt.2007.11.003.

[R6] Arkady Pikovsky, Michael Rosenblum, and Jurgen Kurths. *Synchronization. A universal concept in nonlinear sciences*. Cambridge University Press, 2001. DOI: 10.1017/cbo9780511755743.

[R7] Steven H Strogatz. *Sync: How order emerges from chaos in the universe, nature, and daily life*. Hyperion, 2003.

[R8] "Impulsive stabilization for control and synchronization of chaotic systems: Theory and application to secure communication." In: *IEEE Trans. Circuits Syst. I* 44.10 (1997), pp. 976–988. DOI: 10.1109/81.633887.

[R9] Zhijun Li, Xiaoqing Cao, and Nan Ding. "Adaptive Fuzzy Control for Synchronization of Nonlinear Teleoperators With Stochastic Time-Varying Communication Delays." In: *IEEE Trans. Fuzzy Syst.* 19.4 (Aug. 2011), pp. 745–757. DOI: 10.1109/tfuzz.2011.2143417.

[R10] H. Shokri-Ghaleh and A. Alfi. "Optimal synchronization of teleoperation systems via cuckoo optimization algorithm." In: *Nonlinear Dyn* 78.4 (Aug. 2014), pp. 2359–2376. DOI: 10.1007/s11071-014-1589-5.

[R11] Constance Hammond, Hagai Bergman, and Peter Brown. "Pathological synchronization in Parkinson's disease: Networks, models and treatments." In: *Trends Neurosci.* 30.7 (July 2007), pp. 357–364. DOI: 10.1016/j.tins.2007.05.004.

[R12] F. Dorfler, M. Chertkov, and F. Bullo. "Synchronization in complex oscillator networks and smart grids." In: *Proceedings of the National Academy of Sciences* 110.6 (Jan. 2013), pp. 2005–2010. DOI: 10.1073/pnas.1212134110.

[R13] Donald E Kirk. *Optimal control theory: An introduction*. Courier Corporation, 2012.

[R14] *Numerical Optimization*. Springer-Verlag, 1999. DOI: 10.1007/b98874.

[R15] Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright. *Optimization for Machine Learning*. Cambridge, USA: MIT Press, 2011.

[R16] Victor Becerra. "Optimal control." In: *Scholarpedia* 3.1 (2008), p. 5354. DOI: 10.4249/scholarpedia.5354.

[R17] *Optimal Control Theory*. Springer-Verlag, 2000. DOI: 10.1007/0-387-29903-3.

[R18] *Handbook of Chaos Control*. Wiley-VCH Verlag GmbH & Co. KGaA, Oct. 2007. DOI: 10.1002/9783527622313.

[R19] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[R20] Michael Schmidt and Hod Lipson. "Distilling Free-Form Natural Laws from Experimental Data." In: *Science* 324.5923 (Apr. 2009), pp. 81–85. DOI: 10.1126/science.1165893.

[R21] E.J. Vladislavleva, G.F. Smits, and D. den Hertog. "Order of Nonlinearity as a Complexity Measure for Models Generated by Symbolic Regression via Pareto Genetic Programming." In: *IEEE Trans. Evol. Computat.* 13.2 (Apr. 2009), pp. 333–349. DOI: 10.1109/tevc.2008.926486.

[R22] Markus Quade, Markus Abel, Kamran Shafi, Robert K. Niven, and Bernd R. Noack. "Prediction of dynamical systems by symbolic regression." In: *Phys. Rev. E* 94.1 (July 2016), p. 012214. DOI: 10.1103/physreve.94.012214.

[R23] Hamid Shokri-Ghaleh and Alireza Alfi. "A comparison between optimization algorithms applied to synchronization of bilateral teleoperation systems against time delay and modeling uncertainties." In: *Appl. Soft Comput.* 24 (Nov. 2014), pp. 447–456. DOI: 10.1016/j.asoc.2014.07.020.

[R24] Vladimir Parezanović, Laurent Cordier, Andreas Spohn, Thomas Duriez, Bernd R. Noack, Jean-Paul Bonnet, Marc Segond, Markus Abel, and Steven L. Brunton. "Frequency selection by feedback control in a turbulent shear flow." In: *J. Fluid Mech.* 797 (May 2016), pp. 247–283. DOI: 10.1017/jfm.2016.261.

[R25] Thomas Duriez, Steven L. Brunton, and Bernd R. Noack. *Machine Learning Control – Taming Nonlinear Dynamics and Turbulence*. Springer International Publishing, 2017. DOI: 10.1007/978-3-319-40624-4.

[R26] Daniel A. Wiley, Steven H. Strogatz, and Michelle Girvan. "The size of the sync basin." In: *Chaos* 16.1 (Mar. 2006), p. 015103. DOI: 10.1063/1.2165594.

[R27] Herman H. Goldstine. *A History of the Calculus of Variations from the 17th through the 19th Century*. Springer New York, 1980. DOI: 10.1007/978-1-4613-8106-8.

[R28] William H Press. *Numerical recipes: The art of scientific computing*. 3rd ed. Cambridge university press, 2007.

[R29] Frank L. Lewis, Draguna L. Vrabie, and Vassilis L. Syrmos. *Optimal Control. Lewis/Optimal Control 3e*. John Wiley & Sons, Inc., Jan. 2012. DOI: 10.1002/9781118122631.

[R30] A. E. Bryson (Jr) and Y. Ho. *Applied Optimal Control*. Halsted Press, 1975.

[R31] M. Athans and P. L. Falb. *Optimal Control: An Introduction to the Theory and Its Applications*. Dover Publications, 2006.

[R32] Gene F. Franklin, J. Da Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems*. 7th ed. Pearson, 2014.

[R33] Nichael Lynn Cramer. "A representation for the adaptive generation of simple sequential programs." In: *Proceedings of an International Conference on Genetic Algorithms and the Applications*. Ed. by John J Grefenstette. Psychology Press, 1985, pp. 183–187.

[R34] *Genetic Programming*. Springer Berlin Heidelberg, 1999. DOI: 10.1007/3-540-48885-5.

[R35] Xin-She Yang. "Metaheuristic Optimization." In: *Scholarpedia* 6.8 (2011), p. 11472. DOI: 10.4249/scholarpedia.11472.

[R36] Sean Luke. *Essentials of Metaheuristics*. 2nd ed. Lulu, 2013. URL: http://cs.gmu.edu/%5Ctextasciitilde%20sean/book/metaheuristics/.

[R37] Drew Fudenberg and Jean Tirole. *Game theory*. MIT press, 1991.

[R38] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimisation: NSGA-II." In: *PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*. Springer-Verlag, 2000, pp. 849–858.

[R39] J. Knowles and D. Corne. "The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation." In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. IEEE. DOI: 10.1109/cec.1999.781913.

[R40] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. "SPEA2: Improving the strength Pareto evolutionary algorithm." In: *TIK-Report*. Vol. 103. Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK), 2001. DOI: 10.3929/ethz-a-004284029.

[R41] François-Michel De Rainville, Félix-Antoine Fortin, Marc-André Gardner, Marc Parizeau, and Christian Gagné. "Deap. enabling nimbler evolutions." In: *SIGEVOlution* 6.2 (Feb. 2014), pp. 17–26. DOI: 10.1145/2597453.2597455.

[R42] Stéfan van der Walt, S Chris Colbert, and Gaël Varoquaux. "The NumPy Array: A Structure for Efficient Numerical Computation." In: *Comput. Sci. Eng.* 13.2 (Mar. 2011), pp. 22–30. DOI: 10.1109/mcse.2011.37.

[R43] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001. URL: http://www.scipy.org/ (visited on 04/29/2018).

[R44] Makoto Matsumoto and Takuji Nishimura. "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator." In: *ACM Trans. Model. Comput. Simul.* 8.1 (Jan. 1998), pp. 3–30. DOI: 10.1145/272991.272995.

[R45] SymPy Dev Team. *SymPy: Python library for symbolic mathematics*. 2016. URL: http://www.sympy.org.

[R46] Markus Quade, Julien Gout, and Markus Abel. *glyph - Symbolic Regression Tools*. Version 0.3.5. Jan. 2018. URL: https://github.com/Ambrosys/glyph. DOI: 10.5281/zenodo.1156654.

[R47] Edward N. Lorenz. "Deterministic Nonperiodic Flow." In: *J. Atmos. Sci.* 20.2 (Mar. 1963), pp. 130–141. DOI: 10.1175/1520-0469(1963)020<0130:dnf>2.0.co;2.

[R48] Ed Bullmore and Olaf Sporns. "Complex brain networks: Graph theoretical analysis of structural and functional systems." In: *Nat Rev Neurosci* 10.3 (Feb. 2009), pp. 186–198. DOI: 10.1038/nrn2575.

[R49] Yoshiki Kuramoto. "Lecture Notes in Physics." In: *International Symposium on Mathematical Problems in Theoretical Physics*. Ed. by H. Araki. Vol. 39. New York: Springer, 1975, p. 420.

[R50] Yoshiki Kuramoto. *Chemical Oscillations, Waves, and Turbulence*. Springer Berlin Heidelberg, 1984. DOI: 10.1007/978-3-642-69689-3.

[R51] Leon Cohen. *Time Frequency Analysis: Theory and Applications*. 1st ed. Prentice Hall, 1994.

5 GLYPH: SYMBOLIC REGRESSION TOOLS

by Markus Quade[1], Julien Gout[1], Markus Abel[1]

[1] Ambrosys GmbH, David-Gilly Straße 1, 14469 Potsdam, Germany

This paper has been submitted to Journal of Open Research Software. For typesetting purposes meta-information and the content of the original manuscript have been split. The meta-information can be found in Section B.1.

Abstract

We present Glyph - a package for genetic programming based symbolic regression. Glyph is designed for usage let by numerical simulations let by real world experiments. For experimentalists, glyph-remote provides a separation of tasks: a clear interface splits the genetic programming optimization task from the evaluation of an experimental (or numerical) run. Thus domain experts should be able to employ symbolic regression in their experiments with ease, even if they are not expert programmers. `Python` is ideal for these purposes, hence Glyph is implemented in `Python`.

Keywords— Symbolic Regression, Genetic Programming, Machine Learning Control, MOGP, Python

5.1 INTRODUCTION

Symbolic regression[S1] is an optimization method to find an optimal representation of a function. The method is “symbolic”, because building blocks of the functions, i.e. variables, primitive functions, and operators, are represented symbolically on the computer. Genetic Programming (GP) [S2] can be implemented to find such a function for system identification[S3, S4] or fluid dynamical control[S5, S6]. Glyph is an effort to separate optimization method and optimization task allowing domain-experts without special programming skills to employ symbolic regression in their experiments. We adopt this separation of concerns implementing a client-server architecture; a minimal communication protocol eases its use. Throughout this paper “experiment” is meant as a synonym for any symbolic regression task including a lab-experiment, a numerical simulation or data fitting.

Previous work on system identification and reverse engineering of conservation laws was reported in[S1, S7]. Modern algorithms also include multi objective optimization[S4] and advances like age fitness based genetic programming[S8] or epigenetic local search [S9]. There exist various approaches to the representation of multi IO problems, including stack- or graph-based representations and pointers [S9, S10].

5.2 IMPLEMENTATION AND ARCHITECTURE

Glyph is intended as a lightweight framework to build an application finding an optimal system representation given measurement data. The main application is intended as system control, consequently a control law is determined and returned. Glyph is built on the idea of loose coupling such that dependencies can be released if wanted.

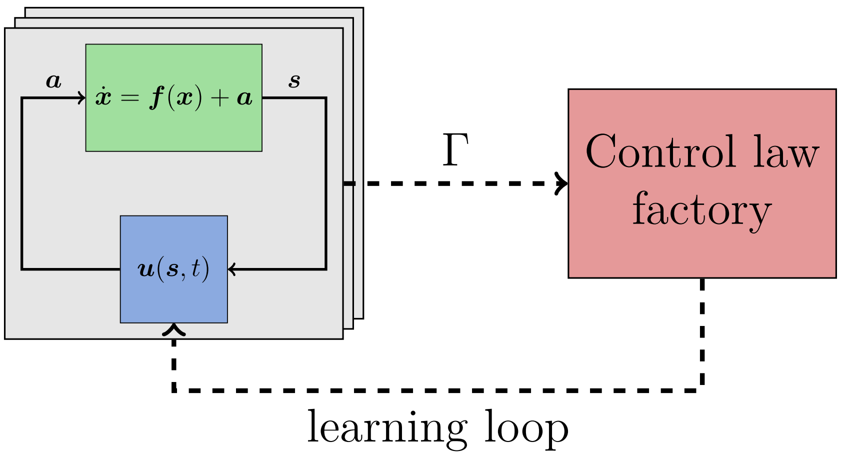

$\boldsymbol{a}$
$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{a}$
$\boldsymbol{s}$
$\boldsymbol{u}(\boldsymbol{s}, t)$
Γ
Control law factory
learning loop

Figure 5.1: Left: A typical closed loop control task is sketched. Given a system $\dot{\vec{x}} = \vec{f}(\vec{x})$, some measurements s and a control law $\vec{u}(\vec{s}, t)$ and we can control the system by adding the actuation $\vec{a} = \vec{u}(\vec{s}, t)$. Right: GP-based symbolic regression finds different candidate control laws. Each candidate solution is given a fitness score Γ which is used to compare different solutions and to advance the search in function space. Figure adapted from [S5] with permission.

A typical control application consists of a system and its controller, possibly separated, *cf.* Fig. 5.1. Glyph has three main abstractions to build such an application: i) the assessment, which holds all methods and data structures belonging to the experiment, ii) the Genetic Programming (GP) which is responsible for the system identification, and iii) the application components, which constitute an application.

5.2.1 *Building an Application*

An application consists of a GP callable, the gp_runner, an assessment callable for input, the assessment_runner, and the application which uses both of these classes and holds all application-relevant details. A command-line application is built by

```
assessment_runner = AssessmentRunner(assess_args)
gp_runner = glyph.application.GPRunner(gp_args)
app = command_line_application(app_args)
```

The assessment_runner has one argument, the parallel_factory which implements a `map()` method, possibly parallel. For an application one needs to implement setup, assign_fitness, and measure: setup is self-explaining, measure is a key method which takes as input a set of measurement functions and combines them into a tuple of callable measures for multi-objective optimization. The measures are used eventually in assign_fitness where the return values are used to assign a fitness to an individual from GP. The interface is freely extensible. A gp_runner forwards the evolutionary iteration. It takes as arguments, gp_args, an individual class, a gp_algorithm, and an assessment_runner. The individual class contains the representation of a function, the individual; it is currently based on Deap's tree-based implementation. The gp_algorithm takes care for the breeding and selection steps, its principles are described in [S2].

The application is run in the main function with `app.run()`. Each of the high-level functions contains a bunch of next-level instructions, and can be built with a minimal assembly of methods.

In the application and gp_runner, the user has freedom to add functionality using the list of callbacks in the arguments, say, to implement other logging or streaming options. This allows for very flexible programming. We constructed the components that way to allow users to specialize for their particular experiments and possibly increase performance or extend the symbolic regression, e. g. by replacing the deap tree-based representation of an individual.

Remote Control

One main objective of glyph is its use in a real experiment. In this case, the GP loop is separated from the experimental loop in a client-server setup using ZeroMQ [S11], *cf.* Fig. 5.2.

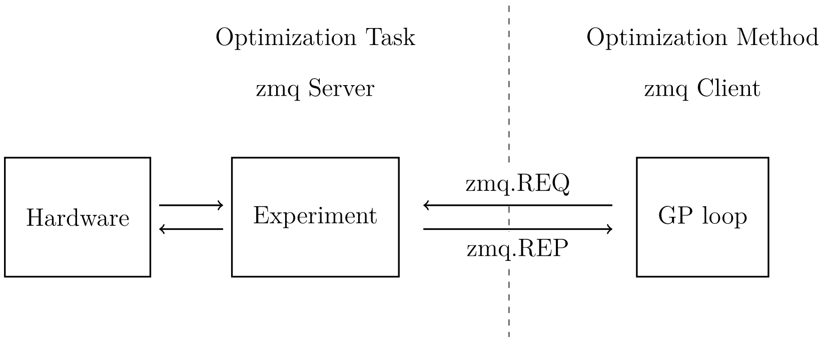


Figure 5.2: Sketch of the implementation of the experiment - GP communication as client-server pattern. Left: single experiment server plus event handler. Right: GP client. Both parts are interfaced using ZeroMQ. As described in Section 5.2.2 the GP program performs requests, e. g. the evaluation of a candidate solution. The event handler takes care of these requests and eventually forwards them to the hardware.

Consequently, one should implement the interface to the experiment using the protocol described in Section 5.2.2. Having the implementation of the experiment, the server, one needs to implement the client, i.e. the interface to the gp_runner. In essence this means connecting the correct sockets with ZeroMQ and ensuring that the gp_runner and the assessment_runner use the corresponding sockets. Then, the main application is assembled as before, now using a *RemoteApp* for the main application, which in turn uses a gp_runner, which then uses now a RemoteAssessmentRunner. That is it, we can run remotely our GP evaluation from some client and the experiment in place of the experiment.

5.2.2 *Communication Protocol*

The communication is encoded in json [S12]. A message is a json object with two members:

```
{
    "action": "value",
    "payload": "value",
}
```

The possible values are listed in Table 5.1. The config action is performed prior

Action name	Payload	Expected return Value
CONFIG	–	config settings
EXPERIMENT	list of expressions	list of fitness value(s)
SHUTDOWN	–	–

Table 5.1: Communication protocol.

to the evolutionary loop. Entering the loop, discovered solutions will be batched and a *experiment* action will be requested. You can configure optional caching for re-discovered solutions. This includes persistent caching between different runs. The *shutdown* action will let the experiment program know that the GP loop is finished and you can safely stop the hardware.

Configuration settings are sent as a json object in key:value form, where the keys contain the option to be set, there is only one mandatory option: the primitive set. To configure the primitive set, the primitive names are passed as content of the key *config*, whose values specify the corresponding arities, both fields described again as json object.

The *experiment* action sends a list of expressions, encoded as string in prefix (also: polish) notation [S13]. For each expression sent, the experiment returns a fitness tuple.

Additionally, one can define the type of algorithm, error metric, representation, hyper-parameters, etc. A comprehensive up to date list can be found at `http://glyph.readthedocs.io/en/latest/usr/glyph_remote/`.

5.2.3 *Application example: control of the chaotic Lorenz System*

In the following, we demonstrate the application and use of Glyph by the determination of an unknown optimal control law for a chaotic system. As an example, we study the control of the potentially chaotic Lorenz system. Chaotic systems are very hard to predict and control in practice due to their sensitivity towards small changes in the initial state which may lead to exponential divergence of trajectories. The Lorenz model [S14] consists of a system of three ordinary differential equations:

$$\begin{aligned} \dot{x} &= s(y - x) \\ \dot{y} &= rx - y - xz \\ \dot{z} &= xy - bz, \end{aligned} \tag{5.1}$$

with two nonlinearities, xy and xz. Here x, y, and z make up the system state and s, r, b are parameters: s is the Prandtl number, r is the Rayleigh number, and b is related to the aspect ratio of the air rolls. For a certain choice of parameters and initial conditions chaotic behavior emerges.

Here we present two examples where the target is to learn control of bring a chaotic Lorenz system to a complete stop, that is, $(x, y, z) = 0$ $(t \in \mathbb{R})$. In the first example, the actuator term is applied to $\dot{y}$. This allows for a more direct control of the system, since y appears in every equation of (5.1) and, thus, influence all three state components, x, y, and z. In the second example the actuator term is applied to $\dot{z}$, which leads to a more indirect control, since the flow of information from z to x is only through y.

Table 5.2: General setup of the GP runs.

population size	500
max. generations	20
MOO algorithm	NSGA-II
tree generation	*halfandhalf*
min. height	1
max. height	4
selection	*selTournament*
tournament size	2
breeding	*varOr*
recombination	*cxOnePoint*
crossover probability	0.5
crossover max. height	20
mutation	*mutUniform*
mutation probability	0.2
mutation max. height	20
constant optimization	*leastsq*

The system setup is summarized in Table 5.2 and Table 5.3. When $r = 28$, $s = 10$, and $b = 8/3$, the Lorenz system produces chaotic solutions (not all solutions are chaotic). Almost all initial points will tend to an invariant set – the Lorenz attractor – a strange attractor and a fractal. When plotted the chaotic trajectory of the Lorenz system resembles a butterfly (blue graph in Fig. 5.3). The target of control is, again, formulated as RMSE of the system state with respect to zero (separately for each component)

$$\Gamma_1 := \mathrm{RMSE}(x, 0), \ \Gamma_2 := \mathrm{RMSE}(y, 0), \ \Gamma_3 := \mathrm{RMSE}(z, 0).$$

The control function u can make use of ideal measurements of the state components. Constant optimization is performed on a single constant k. The respective

Table 5.3: Control of the Lorenz system: system setup.

dynamic system		GP	
s	10	cost functionals	$\text{RMSE}(x, 0)$
r	28		$\text{RMSE}(y, 0)$
b	8/3		$\text{RMSE}(z, 0)$
$x(t_0)$	10.0		$\text{length}(u)$
$y(t_0)$	1.0	argument set	$\{x, y, z\}$
$z(t_0)$	5.0	constant set	$\{k\}$
t_0, t_n	0, 100	seed (in y)	4360036820278701581
n	5000	seed (in z)	2480329230996732981

GP runs for control in y and control in z are conducted with the corresponding random seeds labeled "in y" and "in z".

CONTROL IN y: For control in y the actuator term u is added to the left side of the equation for $\dot{y}$ in the uncontrolled system (5.1):

$$\dot{y} = rx - y - xz + u(x, y, z).$$

The Pareto solutions from the GP run are shown in Table 5.4. The wide spread of the cost indices is a sign of conflicting objectives that are hard to satisfy in conjunction. Interestingly, almost all solutions, u, commonly introduce a negative growth rate into $\dot{y}$. This effectively drives y to zero and suppresses the growth terms, sy and xy, in the equations for $\dot{x}$ and $\dot{z}$ respectively, in turn, driving x and z to zero as well. As would be expected, minimal expressions, of length 1 or 2, cannot compete in terms of the RMSE. For example, the simple solution, $u(x, y, z) = -ky$ (fourth row), is almost as good as the lengthier one, $u(x, y, z) = -\exp(x) + ky$ (first row), and even better in RMSE_y.

Table 5.4 shows the results from the GP run. One solution immediately stands out: $u = k \cdot x + z$, with $k = -27.84$ (second row). It is exactly what one might expect as a control term for the chaotic Lorenz system with control in y. This control law effectively reduces the Rayleigh number r to a value close to zero ($k \approx r$), pushing the Lorenz system past the first pitchfork bifurcation, at $r = 1$, back into the stable-origin regime. If $r < 1$ then there is only one equilibrium point, which is at the origin. This point corresponds to no convection. All orbits converge to the origin, which is a global attractor, when $r < 1$.

The phase portrait of the solution from the first and second row of Table 5.4 are illustrated in Fig. 5.3. After a short excursion in negative y direction ($t \approx 5$), the green trajectory quickly converges to zero. The red trajectory seems to take a shorter path in phase space, but, it is actually slower to converge to the origin. This is verified by a plot of the trajectories for the separate dimensions x, y and z over time Fig. 5.4.

Table 5.4: Control of the Lorenz system in y: Pareto-front solutions.

RMSE_x	RMSE_y	RMSE_z	length	expression	constants
0.178884	0.087476	0.105256	7	$-\exp(x) + k \cdot y$	$k = -135.43$
0.241226	0.069896	0.213063	5	$k \cdot x + z$	$k = -27.84$
0.246315	0.014142	0.222345	6	$-z + k \cdot y$	$k = -75590.65$
0.246316	0.014142	0.222347	4	$-k \cdot y$	$k = 75608.50$
0.246367	0.028851	0.220426	10	$-x \cdot (k + y) \cdot \exp(\exp(y))$	$k = 9.62$
0.246729	0.118439	0.211212	6	$-x \cdot (k + y)$	$k = 29.21$
0.246850	0.031747	0.220726	9	$-x \cdot (k + y) \cdot \exp(y)$	$k = 26.12$
4.476902	4.468534	7.488516	3	$-\exp(y)$	
7.783655	8.820086	24.122441	2	$-x$	
7.931978	9.066296	25.047630	1	k	$k = 1.0$
8.319191	8.371462	25.932887	2	$-y$	
8.994685	9.042226	30.300641	1	z	

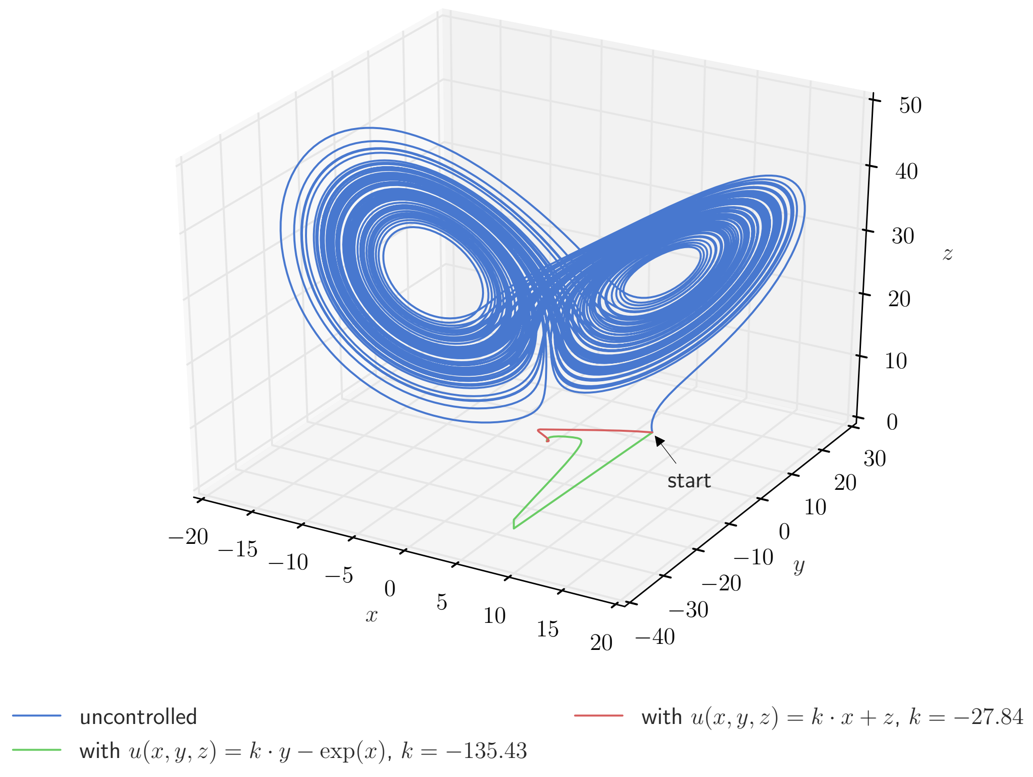


Figure 5.3: Phase portrait of the forced Lorenz system with control exerted in $\dot{y}$. (Green and red: The system trajectories when controlled by two particular Pareto-front solutions. Blue: the uncontrolled chaotic system.)

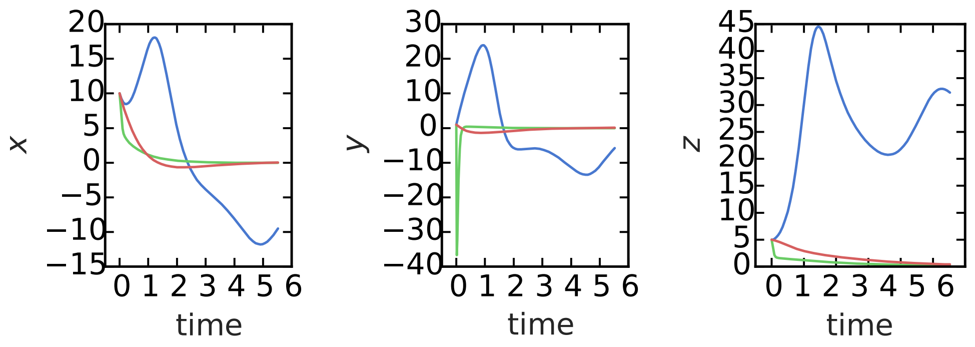


Figure 5.4: Detailed view of the single trajectories in x, y, and z dimension. (blue: uncontrolled; green: $u(x,y,z) = -\exp(x) + k \cdot y$, $k = -135.43$; red: $u(x,y,z) = k \cdot x + z$, $k = -27.84$.)

CONTROL IN z: For control in z the actuator term u is added to the left side of the equation for $\dot{z}$ in the uncontrolled system (5.1)

$$\dot{z} = xy - bz + u(x,y,z)$$

Selected Pareto-front individuals from the GP run are displayed in Table 5.5. As mentioned at the beginning of this section, effective control is hindered by the indirect influence of z on the other state variables, hence, it is not surprising that the control laws here are more involved than in the previous case. Also, they generally do not perform well in the control of z, which is expressed by the relatively high values in RMSE_z. This is confirmed by the phase portrait of the solution $u(x,y,z) = -(k \cdot (-y) + x \cdot z + y + z)$ shown in figure Fig. 5.5: While going straight to the origin in the xy-plane there are strong oscillations of the trajectory along the z-axis.

The dynamics caused by the actuation, e. g. for the best control law found, can be explained qualitatively: there is a strong damping in all variables but y. This reflects the tendency to suppress z-oscillations and, at the same time, to add damping in y through the xz term: if y grows, the z contribution to damping on the right hand side of the Lorenz equations (5.1) grows and, in turn, damps y. This is, however, only possible to some extent, hence, the oscillations observed in figure Fig. 5.5.

We conclude the demonstration with a short summary: Using Glyph we can find complex control laws, even for unknown systems. This cannot be easily achieved with other frameworks. The control laws found can be studied analytically in contrast to several other methods which have black-box character. The usage is straightforward, as we have described above. The above example can be found online as an example.

Other symbolic regression libraries

Due to its popularity, symbolic regression is implemented by most genetic programming libraries. A semi-curated list can be found at `http://geneticprogramming.com/software/`. In contrast to other implementations, Glyph implements higher concepts, such as symbolic constant optimization, and also offers parallel execu-

Table 5.5: Control of the Lorenz system in z: selected Pareto-front solutions.

RMSE_x	RMSE_y	RMSE_z	length	expression	constants
0.289289	0.139652	26.994070	13	$-(k \cdot (-y) + x \cdot z + y + z)$	$k = 793.129676$
0.327926	0.267043	27.070289	8	$\exp(-k + y \cdot \sin(y))$	$k = -4.254574$
0.431993	0.508829	32.116326	7	$(k + x) \cdot (y + z)$	$k = 2.638069$
0.471535	0.525010	26.986321	5	$k + x \cdot z$	$k = 67.137183$
0.637056	0.605686	26.895493	7	$\exp(k + y \cdot \sin(y))$	$k = 3.964478$
0.677204	0.703577	27.019308	4	$y + \exp(k)$	$k = 4.276256$
0.930668	0.952734	26.895126	5	$x + \exp(\exp(k))$	$k = 1.448198$
1.764030	1.860288	26.766383	6	$(k + x) \cdot \exp(y)$	$k = 21.783557$

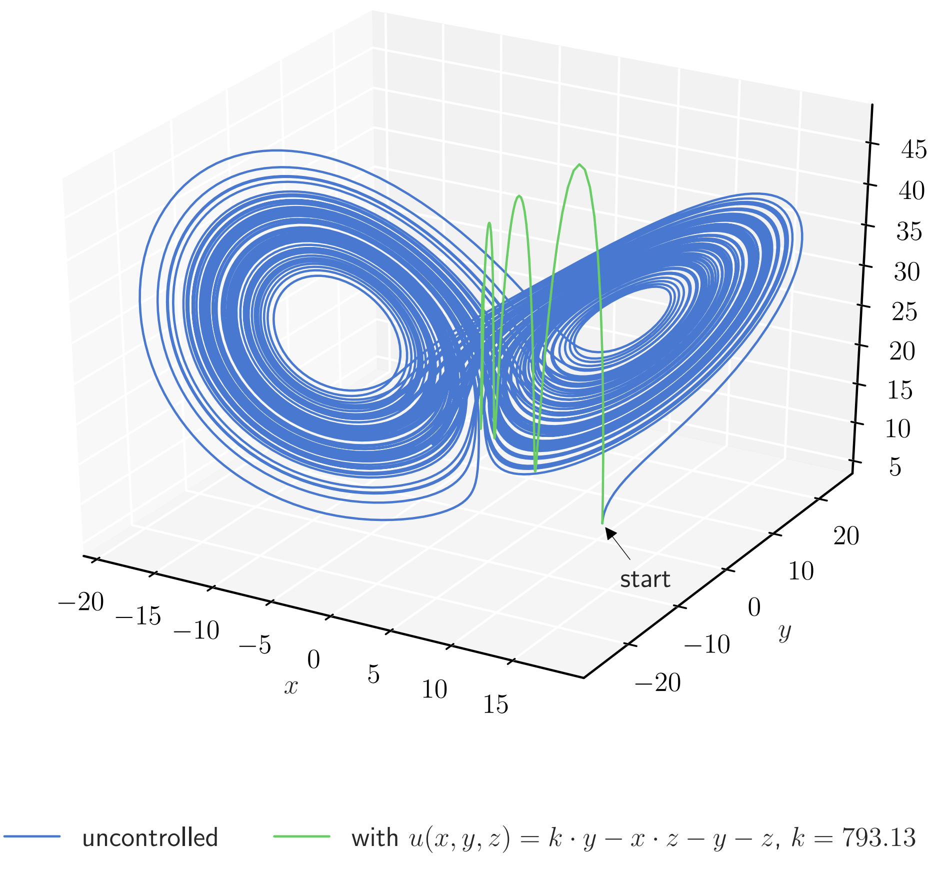


Figure 5.5: Control of the Lorenz system in $\dot{z}$.

tion for complex examples (control simulation, system identification). Glyph is well tested, *cf.* Table 5.6 and currently applied in two experiments and several numerical problems. For control, there exists a dedicated `Matlab` toolbox (with `Python` interface), openMLC [S15], which contains much of the material treated in [S6].

	CI/tests	doc	open api	caching	CP	MOGP	SCO	MO
openMLC	✓	✗	✗	✓	✓	✗	✗	✗
Glyph	✓	✓	✓	✓	✓	✓	✓	✓

Table 5.6: Comparison of `Glyph` and `openMLC` features. MOGP refers to multi objective optimization. CP refers to checkpointing. MO means multiple outputs. SCO means symbolic constant optimization.

5.3 REUSE POTENTIAL

The potential to use `Glyph` is twofold: one one hand applications can be easily written and the elegant core functionality can be extended; on the other hand, researchers can use the code as core for symbolic regression and extend its functionality in a very generic way. With respect to applications, currently two main directions are targeted: modeling using genetic programming- based symbolic regression and the control of complex system, where a control law can be found generically, using genetic programming. The detailed examples and tutorial allow usage from beginner to experienced level, i.e. undergraduate research projects to faculty research. The design of Glyph is such that generic interfaces are provided allowing for very flexible extension.

ACKNOWLEDGEMENTS

We acknowledge very fruitful discussions with respect to applications of MLC with S. Brunton, B. Noack, A. Pikovsky, M. Rosenblum, R. Semaan, and B. Strom and coding gossip with V. Mittal.

FUNDING STATEMENT

This work has been partially supported by the German Science Foundation via SFB 880. MQ was supported by a fellowship within the FITweltweit program of the German Academic Exchange Service (DAAD).

REFERENCES

[S1] Michael Schmidt and Hod Lipson. "Distilling Free-Form Natural Laws from Experimental Data." In: *Science* 324.5923 (Apr. 2009), pp. 81–85. DOI: 10.1126/science.1165893.

[S2] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[S3] Ekaterina Vladislavleva, Tobias Friedrich, Frank Neumann, and Markus Wagner. "Predicting the energy output of wind farms based on weather data: Important variables and their correlation." In: *Renewable Energy* 50 (Feb. 2013), pp. 236–243. DOI: 10.1016/j.renene.2012.06.036.

[S4] Markus Quade, Markus Abel, Kamran Shafi, Robert K. Niven, and Bernd R. Noack. "Prediction of dynamical systems by symbolic regression." In: *Phys. Rev. E* 94.1 (July 2016), p. 012214. DOI: 10.1103/physreve.94.012214.

[S5] Julien Gout, Markus Quade, Kamran Shafi, Robert K. Niven, and Markus Abel. "Synchronization control of oscillator networks using symbolic regression." In: *Nonlinear Dyn* 91.2 (Nov. 2017), pp. 1001–1021. DOI: 10.1007/s11071-017-3925-z.

[S6] Thomas Duriez, Steven L. Brunton, and Bernd R. Noack. *Machine Learning Control – Taming Nonlinear Dynamics and Turbulence*. Springer International Publishing, 2017. DOI: 10.1007/978-3-319-40624-4.

[S7] Michael D Schmidt, Ravishankar R Vallabhajosyula, Jerry W Jenkins, Jonathan E Hood, Abhishek S Soni, John P Wikswo, and Hod Lipson. "Automated refinement and inference of analytical models for metabolic networks." In: *Phys. Biol.* 8.5 (Aug. 2011), p. 055011. DOI: 10.1088/1478-3975/8/5/055011.

[S8] Michael Schmidt and Hod Lipson. "Age-Fitness Pareto Optimization." In: *Genetic Programming Theory and Practice VIII*. Vol. 8. Springer New York, Oct. 2010, pp. 129–146. DOI: 10.1007/978-1-4419-7747-2_8.

[S9] William La Cava, Kourosh Danai, and Lee Spector. "Inference of compact nonlinear dynamic models by epigenetic local search." In: *Eng. Appl. Artif. Intell.* 55 (Oct. 2016), pp. 292–306. DOI: 10.1016/j.engappai.2016.07.004.

[S10] Edgar Galvan-Lopez. "Efficient graph-based genetic programming representation with multiple outputs." In: *Int. J. Autom. Comput.* 5.1 (Jan. 2008), pp. 81–89. DOI: 10.1007/s11633-008-0081-4.

[S11] Faruk Akgul. *ZeroMQ*. Packt Publishing, 2013.

[S12] Ecma International. "The JSON Data Interchange Format." In: *Standard ECMA-404* 9 (2013).

[S13] G. Jorke, B. Lampe, and N. Wengel. *Arithmetische Algorithmen der Mikrorechentechnik*. Verlag Technik, 1989.

[S14] Edward N. Lorenz. "Deterministic Nonperiodic Flow." In: *J. Atmos. Sci.* 20.2 (Mar. 1963), pp. 130–141. DOI: 10.1175/1520-0469(1963)020<0130:dnf>2.0.co;2.

[S15] MachineLearningControl. *OpenMLC-Python*. Aug. 2017. URL: https://github.com/MachineLearningControl/OpenMLC-Python (visited on 04/29/2018).

6 CONCLUSIONS

In this thesis, we have demonstrated the use of symbolic regression in the natural sciences with application to identification, prediction and control of dynamical system. Traditionally, dynamical systems modeling relies on analytic or numeric methods and is conducted by domain experts. Machine Learning (ML), and in particular symbolic regression, offers a new paradigm: *Data-driven* modeling. ML often provides only *black box models*, *i. e.* models which lack insight and interpretability for domain experts. Similarly, the origins of data are also obscure for most ML experts.

The No Free Lunch Theorems (NFL) [87, 88] state that all methods[1] of finding solutions for optimization problems perform equally well if averaged over all possible optimization problems. No single solution-finding method performs best across all types of problems; in other words, each method can outperform other methods in a specific domain. While the domain of some methods, e. g. deep learning, is huge, generating a lot of commercial profit and media attention, symbolic regression certainly has its place close to scientific applications.

In contrast to most ML-methods, symbolic regression provides *white-box models* which allow for deep insight. While interpretability increases scientists' trust in the optimization method itself, it also allows them to stick to their conventional toolchain. In the case of dynamical systems, this includes for example stability analysis, model integration, time-scale separation or series expansion. Most importantly, scientists may also be able to use their intuition, e. g. associating certain expressions with patterns in phase-space.

We introduce two kinds of symbolic regression: sparse regression-based and Genetic Programming (GP)-based symbolic regression. They are applied to system identification, prediction and control problems.

The focus of this thesis is the further advancement of a methodological foundation of symbolic regression and its application to dynamical systems. For system identification tasks, both kinds of symbolic regression methods have been applied successfully. However, if one has a good understanding of the function space, sparse regression offers a more robust estimator with faster convergence. Additionally we have introduced the concept of fast model recovery, relying on first principles of dynamical systems. With increasingly abstract objectives or complex systems, GP-based symbolic regression starts to shine. Gradients on abstract complex objectives are often intractable, thus evolutionary optimization will be more efficient. Furthermore, GP inherently supports multiple outputs as well as multiple objectives which are both of great value.

We applied symbolic regression to a range of current problems in dynamical systems and data science ranging from predicting the power production of green energy power plants (wind or solar) and front propagation in coupled oscillators,

1 Note, that this includes random search as well.

to turbulence control around an airfoil or crossflow turbines, and synchronization control in coupled oscillators.

We have motivated the sparsity or parsimony principle in symbolic regression. In sparse regression, it is achieved using regularization and thresholding. The principle is also eponymous for sparse regression. In GP-based symbolic regression we use multi-objective optimization by introducing a complexity measure as a second objective. The accuracy-simplicity trade-off typically gives rise to a set of Pareto-optimal models which requires further inspection to select a single model. Multi-objectivity has been proven to be a vital component in GP combating bloat and overfitting [89]. It is of further use beyond attaining sparsity if the objective is complex and the trade-off between different aspects is unclear. For time series prediction, there is often a non-linear cost function associated to the residual errors between forecast and actual event. For example, predicting the power output of green energy power plants, this could mean earning less in the case of under-prediction, as the under-predicted amount will have a lower price per kWh. However, if one promises too much, conventional energy resources have to be bought which leads to diminishing returns. If the exact values and prices are known, such a scenario can be expressed in a single objective. However, since these parameters depend on time it might be useful to minimize over- and under-prediction independently without formulating the exact trade-off. The same principle holds for turbulence control minimizing drag while maximizing lift.

A significant part of the contributions of this thesis consists of accompanying scientific open source software, *cf.* Appendix B. For sparse regression, several previously scattered methods have been made available in [90] adhering to the established `scikit-learn` interface [91]. There are many implementations of GP available already[2]. Most of them are conceived as frameworks setting a huge entry barrier for users. With `Glyph` [92], we try to lower this entry barrier. `Glyph` was developed with MLC in mind and is used at the Collaborative Research Center (CRC) 880 in Braunschweig [93].

OUTLOOK

Motif discovery will be an interesting topic in the future. In symbolic regression, a motif is a recurring parametrized pattern, e. g. the Michaelis–Menten kinetics reaction rate occurring in the quasi steady state approximation of the chemical Master equation [94]. In different communities, different approaches have been developed so far: on the one hand, specialized models [95] are used in natural sciences while on the other hand, clever structure representations [15] are used in the ML community, *cf.* Section A.1.

Typically, within each domain there is a catalog of these motifs as well as a graphical representation to visually encode structure [96, 97]. Graphical representation allows for a more compact encoding while maintaining the symbolic character which ultimately increases the inference ability of domain experts, as well as their ability to communicate the results to other humans. In systems biology for example, changing the type of a neuron or coupling between them is a single symbolic change which has a huge impact on the underlying dynamics. GP-based

2 See also `http://geneticprogramming.com/software/`.

symbolic regression is a perfect candidate if primitives encode motifs instead of mathematical functions.

In traditional mathematics, if a problem occurs often enough, we usually introduce a special function and associate it with the solution of that problem, e. g. the exponential function as a solution of $\dot{x} = x$. Often, only approximate numerical evaluation is possible and the *exact* implementation on computing devices is hidden from the user. Similarly, we can understand motifs as symbolized solutions to often occuring problems, e. g. the propagator of the Lorenz system or a global mean field coupling term for coupled oscillators.

Using meta-data and side-information may be critical in ML tasks [98]. Metadata may aid in the choice of building blocks, *i. e.* the primitive set in GP or the candidate library in sparse regression. The correct choice is critical for the success and often a big criticism of the method.

We could use symbolic regression to find motifs, but also *any other ML-method*. On a higher level of calculus, implementation details only affect the precision of a solution, not its accuracy. Motifs would inform knowledge transfer between progressively more difficult optimization problems and symbolic regression can be the framework supporting motif calculus, making GP-based symbolic regression a language for optimization itself.

The design of model falsifying experiments is another key component of human expert modeling and its automation will be tremendously beneficial for symbolic regression. A good model not only explains current experiments, but also predicts the future accurately. Modeling complex phenomena is always a learning process, starting from understanding simple edge cases to an eventual deep understanding of the problem. Currently, this learning process is guided by domain experts with their ability to formulate theories and design experiments to verify those theories. Teaching machines how to learn will require to teach them how to verify and falsify models as this is an essential skill to estimate the boundaries of knowledge and the trust them. Co-evolution [99] in GP-based symbolic regression points towards that vein of research. In co-evolution, the fitness function is evolved alongside the candidate solution ensuring maximal discriminability. Stage-wise model refinement is at the core of science. Expanding on the idea of co-evolution, automated experiment design should lead to progressively more challenging problems. The knowledge transfer between the stages emulates learning, eventually enabling a knowledge exchange between domains.

A higher incorporation in the modeling process of the aforementioned strength of symbolic regression, namely proving scientific model insight via mathematical expressions, will be of big benefit. Current efforts include automatic differentiation [100, 101] which has a big impact on constant optimization reducing the number of necessary function calls significantly, a problem the MLC community is struggling with, *cf.* Appendix C. Incorporating automatic differentiation in symbolic regression based control could increase practicality of this approach. Another viewpoint, partially discussed in Chapter 2, is automated stability analysis and Lyapunov exponent calculation. The former can be used for further refinement of the model selection process in control tasks, e. g. formulating a stability threshold as an objective or constraint. Symbolic models may be used by tools like Auto [102] which reduce the integration problem to an algebraic one, which is again useful for constant optimization. Lyapunov exponents can independently be calculated from data and thus their comparison with the model estimation offers great insight and aids model selection as well.

The work on hybrid methods is promising, e. g. combining sparse regression and GP [51, 52]. First attempts have also been made to optimize the kernel function of a Gaussian process [103]. A next logical step seems to be the symbolic regression aided discovery of Koopman observables, *i. e.* a set of measurement functions which evolve via the linear Koopman operator forward in time.

Traditional ML methods allow for easy combination via pipelines [104], e. g. for feature engineering, hyper-parameter estimation or ensemble estimators. While this is essential for the success of ML, pipelining is conducted under the black-box assumption. Currently, using symbolic regression like a traditional ML method means giving up on interpretability and other perks of the white-box symbolic models. More research has to be conducted to bring these two ideas together. One approach could be the automated optimization of embedded coordinates, a technique which has been successfully used before in a non automated form in symbolic regression [41, 105].

Looking at the current shooting star of ML – deep learning – there is huge potential for cross-fertilization. Recurrent Neural Networks look promising for modeling time dependencies in time series prediction and system identification. Recursion allows for compact encoding of algorithms and may be useful in GP-based symbolic regression allowing for models with time delays. Encoding time delays as primitives in GP could be valuable in describing automated embedding in time series prediction models. Time delays also often occur in coupled systems [106], and modeling them can have a huge impact on the stability analysis regarding the control of those systems [107].

The implementation of constraints is still an open question and of great interest in symbolic regression, especially GP-based symbolic regression. This is relevant in real-world applications where each function evaluation has an associated cost. In an overly constrained search-space, this is very inefficient and the evolutionary optimization is close to random search. Local search techniques may help to find constraint conforming corrections [36]. Currently, candidate solutions are rejected based on a heuristic, e. g. by comparison to current best results, mathematical properties or output-statistics based on typical inputs. Typically, a rejected candidate solution or a rejected modification of a candidate solution is simply replaced by a new one.

Finally, the successful application of symbolic regression to real world problems continues to be an exciting challenge: obvious applications include time series prediction for commercial profits, e. g. in the financial sector. This seems to be a popular exit strategy for former academics [108, 109]. For commercial applications, incorporating meta- and side-information, e. g. large-scale weather pattern for forecasting the power production of wind turbines, will be extremely valuable. MLC had a great initial success and needs to be expanded to more realistic scenarios, including stability analysis of control laws. The idea of synchronization control which has been investigated in Chapter 4 can be expanded to deep brain stimulation for tremor or epilepsy treatment [110]. As a relatively young optimization method, there is engineering work left to do to increase real-world practicability, *i. e.* increasing robustness by reducing the number of hyper-parameters as well as designing more user friendly interfaces.

BIBLIOGRAPHY

[1] Amy Brand, Liz Allen, Micah Altman, Marjorie Hlava, and Jo Scott. "Beyond authorship: Attribution, contribution, collaboration, and credit." In: *Learn. Pub.* 28.2 (Apr. 2015), pp. 151–155. DOI: 10.1087/20150211.

[2] M. Hilbert and P. Lopez. "The World's Technological Capacity to Store, Communicate, and Compute Information." In: *Science* 332.6025 (Feb. 2011), pp. 60–65. DOI: 10.1126/science.1200970.

[3] European Climate Research Alliance. URL: http://www.ecra-climate.eu/ (visited on 04/29/2018).

[4] Ecmwf. *IFS Documentation CY41R1*. IFS Documentation. Reading, England: ECMWF, 2015.

[5] Collaborative Research Center 880. *Fundamentals of High Lift of Future Civil Aircraft*. URL: https://www.tu-braunschweig.de/sfb880 (visited on 04/29/2018).

[6] *Self-Driving Car Project*. URL: https://waymo.com/ (visited on 04/29/2018).

[7] European Union. *Human Brain Project*. URL: https://www.humanbrainproject.eu/ (visited on 04/29/2018).

[8] The National Academies of Science, Engineering, and Medicine. *Transportation Research Board*. URL: http://www.trb.org/ (visited on 04/29/2018).

[9] Boris S. Kerner. *The Physics of Traffic*. Springer Berlin Heidelberg, 2004. DOI: 10.1007/978-3-540-40986-1.

[10] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. "Congested traffic states in empirical observations and microscopic simulations." In: *Phys. Rev. E* 62.2 (Aug. 2000), pp. 1805–1824. DOI: 10.1103/physreve.62.1805.

[11] Siam. *Coference Archives and Future Meetings*. URL: https://www.siam.org/meetings/archives.php#DS (visited on 04/29/2018).

[12] Uriel Frisch. *Turbulence: The legacy of AN Kolmogorov*. Cambridge university press, 1995.

[13] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000. DOI: 10.1017/cbo9780511840531.

[14] Kerson Huang. *Introduction to statistical physics*. CRC Press, 2009.

[15] Michael D Schmidt, Ravishankar R Vallabhajosyula, Jerry W Jenkins, Jonathan E Hood, Abhishek S Soni, John P Wikswo, and Hod Lipson. "Automated refinement and inference of analytical models for metabolic networks." In: *Phys. Biol.* 8.5 (Aug. 2011), p. 055011. DOI: 10.1088/1478-3975/8/5/055011.

[16] Theodore Wilbur Anderson. *An introduction to multivariate statistical analysis*. Vol. 2. Wiley New York, 1958.

[17] Vilfredo Pareto. *Manuel d'économie politique*. Vol. 38. V. Giard & E. Brière, 1909.

[18] Pedro Domingos. "A few useful things to know about machine learning." In: *Commun. ACM* 55.10 (Oct. 2012), p. 78. DOI: 10.1145/2347736.2347755.

[19] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. "Feature Selection. A Data Perspective." In: *CSUR* 50.6 (Dec. 2017), pp. 1–45. DOI: 10.1145/3136625.

[20] Trevor Hastie, Jerome Friedman, and Robert Tibshirani. *The Elements of Statistical Learning*. Vol. 2. 1. Springer New York, 2001. DOI: 10.1007/978-0-387-21606-5.

[21] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer New York, 2013. DOI: 10.1007/978-1-4614-7138-7.

[22] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems." In: *Proc Natl Acad Sci USA* 113.15 (Mar. 2016), pp. 3932–3937. DOI: 10.1073/pnas.1517384113.

[23] Trent McConaghy. "FFX: Fast, Scalable, Deterministic Symbolic Regression Technology." In: *Genetic and Evolutionary Computation*. Springer New York, 2011, pp. 235–260. DOI: 10.1007/978-1-4614-1770-5_13.

[24] Robert Tibshirani. "Regression shrinkage and selection via the lasso: A retrospective. Regression Shrinkage and Selection via the Lasso." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.3 (Apr. 2011), pp. 273–282. DOI: 10.1111/j.1467-9868.2011.00771.x.

[25] Andrey Tikhonov. "Solution of incorrectly formulated problems and the regularization method." In: *Soviet Meth. Dokl.* 4 (1963), pp. 1035–1038.

[26] Hui Zou and Trevor Hastie. "Regularization and variable selection via the elastic net." In: *J Royal Statistical Soc B* 67.2 (Apr. 2005), pp. 301–320. DOI: 10.1111/j.1467-9868.2005.00503.x.

[27] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "A Sparse-Group Lasso." In: *Journal of Computational and Graphical Statistics* 22.2 (Apr. 2013), pp. 231–245. DOI: 10.1080/10618600.2012.681250.

[28] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *A note on the group lasso and a sparse group lasso*. 2010. eprint: 1001.0736 (math.ST).

[29] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems." In: *Proc Natl Acad Sci USA* 113.15 (Mar. 2016), pp. 3932–3937. DOI: 10.1073/pnas.1517384113.

[30] Kristofer E Bouchard. *Bootstrapped Adaptive Threshold Selection for Statistical Model Selection and Estimation*. May 2015. arXiv: 1505.03511 [stat.ML].

[31] Leo Breiman and Jerome H. Friedman. "Estimating Optimal Transformations for Multiple Regression and Correlation." In: *J. Am. Stat. Assoc.* 80.391 (Sept. 1985), pp. 580–598. DOI: 10.1080/01621459.1985.10478157.

[32] Pauline Elma Clara Claussen. "Regression: When a Nonparametric Approach is Most Fitting." Master's Thesis. The University of Texas, Austin, May 2012.

[33] John Fox. *Multiple and Generalized Nonparametric Regression*. 131. SAGE Publications, Inc., 2000. DOI: 10.4135/9781412985154.

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

[35] Charles Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. 1859.

[36] Sean Luke. *Essentials of Metaheuristics*. 2nd ed. Lulu, 2013. URL: http://cs.gmu.edu/%5Ctextasciitilde%20sean/book/metaheuristics/.

[37] J. Bongard and H. Lipson. "Automated reverse engineering of nonlinear dynamical systems." In: *Proceedings of the National Academy of Sciences* 104.24 (June 2007), pp. 9943–9948. DOI: 10.1073/pnas.0609476104.

[38] William G. La Cava and Kourosh Danai. "Gradient-based adaptation of continuous dynamic model structures." In: *Int. J. Syst. Sci.* 47.1 (Aug. 2015), pp. 249–263. DOI: 10.1080/00207721.2015.1069905.

[39] Sean Stijven, Ekaterina Vladislavleva, Arthur Kordon, Lander Willem, and Mark E Kotanchek. "Genetic Programming Theory and Practice XIII." In: (2016), pp. 241–260. DOI: 10.1007/978-3-319-34223-8.

[40] William La Cava, Kourosh Danai, Lee Spector, Paul Fleming, Alan Wright, and Matthew Lackner. "Automatic identification of wind turbine models using evolutionary multiobjective optimization." In: *Renewable Energy* 87.October 2015 (Mar. 2016), pp. 892–902. DOI: 10.1016/j.renene.2015.09.068.

[41] Markus Quade, Markus Abel, Kamran Shafi, Robert K. Niven, and Bernd R. Noack. "Prediction of dynamical systems by symbolic regression." In: *Phys. Rev. E* 94.1 (July 2016), p. 012214. DOI: 10.1103/physreve.94.012214.

[42] Thomas Duriez, Steven L. Brunton, and Bernd R. Noack. *Machine Learning Control – Taming Nonlinear Dynamics and Turbulence*. Vol. 116. Fluid Mechanics and Its Applications. Springer International Publishing, 2017. DOI: 10.1007/978-3-319-40624-4.

[43] Markus Quade, Julien Gout, and Markus Abel. "Glyph: Symbolic Regression Tools." In: *J Open Res Softw* (Sept. 2017). arXiv: 1803.06226 [cs.MS]. Submitted.

[44] Julien Gout, Markus Quade, Kamran Shafi, Robert K. Niven, and Markus Abel. "Synchronization control of oscillator networks using symbolic regression." In: *Nonlinear Dyn* 91.2 (Nov. 2017), pp. 1001–1021. DOI: 10.1007/s11071-017-3925-z.

[45] Michael Schmidt and Hod Lipson. "Distilling Free-Form Natural Laws from Experimental Data." In: *Science* 324.5923 (Apr. 2009), pp. 81–85. DOI: 10.1126/science.1165893.

[46] Kenneth O. Stanley and Risto Miikkulainen. "Evolving Neural Networks through Augmenting Topologies." In: *Evol. Comput.* 10.2 (June 2002), pp. 99–127. DOI: 10.1162/106365602320169811.

[47] Riccardo Poli and William B Langdon. "Sub-machine-code Genetic Programming." In: *Advances in genetic programming* 3 (1999), p. 301.

[48] Christopher Fusting. "Temporal Feature Selection with Symbolic Regression." Master's Thesis. University of Vermont, Oct. 2017. URL: http://scholarworks.uvm.edu/graddis/806.

[49] Christopher Fusting, Marcin Szubert, Josh Bongard, and Christian Skalka. "Finding Temporal Features with Symbolic Regression." In: *Proceedings of the 2018 on Genetic and Evolutionary Computation Conference - GECCO '18*. Kyoto, Japan: ACM press, July 2018.

[50] Ilknur Icke and Joshua C. Bongard. "Improving genetic programming based symbolic regression using deterministic machine learning." In: *2013 IEEE Congress on Evolutionary Computation*. IEEE, June 2013, pp. 1763–1770. DOI: 10.1109/cec.2013.6557774.

[51] Ignacio Arnaldo, Una-May O'Reilly, and Kalyan Veeramachaneni. "Building Predictive Models via Feature Synthesis." In: *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15*. New York, New York, USA: ACM Press, 2015, pp. 983–990. DOI: 10.1145/2739480.2754693.

[52] William La Cava and Jason Moore. "A General Feature Engineering Wrapper for Machine Learning Using epsilon-Lexicase Survival." In: *Lecture Notes in Computer Science*. Vol. 17. 3. Springer International Publishing, Oct. 2017, pp. 80–95. DOI: 10.1007/978-3-319-55696-3_6.

[53] Michael Kommenda, Gabriel Kronberger, Stephan Winkler, Michael Affenzeller, and Stefan Wagner. "Effects of constant optimization by nonlinear least squares minimization in symbolic regression." In: *Proceeding fifteenth Annu. Conf. companion Genet. Evol. Comput. Conf. companion - GECCO '13 Companion* (2013), p. 1121. DOI: 10.1145/2464576.2482691.

[54] Jan Žegklitz and Petr Pošík. *Learning Linear Feature Space Transformations in Symbolic Regression*. 2017. arXiv: 1704.05134 [cs.NE].

[55] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer US, 1971. DOI: 10.1007/978-1-4615-7566-5.

[56] Karl Pearson. "On lines and planes of closest fit to systems of points in space." In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572.

[57] F. Takens. *Detecting strange attractors in turbulence*. Lecture Notes in Mathematics No.898. Berlin: Springer, 1981.

[58] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. "Efficient and Robust Automated Machine Learning." In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pp. 2962–2970. URL: http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf.

[59] Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. "Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science." In: *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference - GECCO '16*. GECCO '16. Denver, Colorado, USA: ACM Press, 2016, pp. 485–492. DOI: 10.1145/2908812.2908918.

[60] Charles Babbage. *Passages from the Life of a Philosopher*. Cambridge University Press, 2009. DOI: 10.1017/cbo9781139103671.

[61] Wikipedia. *Garbage in, garbage out — Wikipedia, The Free Encyclopedia*. 2018. URL: http://en.wikipedia.org/w/index.php?title=Garbage%5C%20in%5C%2C%5C%20garbage%5C%20out%5C&oldid=830203452 (visited on 04/29/2018).

[62] B. O. Koopman. "Hamiltonian Systems and Transformation in Hilbert Space." In: *Proceedings of the National Academy of Sciences* 17.5 (May 1931), pp. 315–318. DOI: 10.1073/pnas.17.5.315.

[63] B. O. Koopman and J. v. Neumann. "Dynamical Systems of Continuous Spectra." In: *Proceedings of the National Academy of Sciences* 18.3 (Mar. 1932), pp. 255–263. DOI: 10.1073/pnas.18.3.255.

[64] Igor Mezić. "Spectral Properties of Dynamical Systems, Model Reduction and Decompositions." In: *Nonlinear Dyn* 41.1-3 (Aug. 2005), pp. 309–325. DOI: 10.1007/s11071-005-2824-x.

[65] Igor Mezić. "Analysis of Fluid Flows via Spectral Properties of the Koopman Operator." In: *Annu. Rev. Fluid Mech.* 45.1 (Jan. 2013), pp. 357–378. DOI: 10.1146/annurev-fluid-011212-140652.

[66] Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Koopman Invariant Subspaces and Finite Linear Representations of Nonlinear Dynamical Systems for Control." In: *PLoS ONE* 11.2 (Feb. 2016), e0150171. DOI: 10.1371/journal.pone.0150171.

[67] Amit Surana and Andrzej Banaszuk. “Linear observer synthesis for nonlinear systems using Koopman Operator framework.” In: *IFAC-PapersOnLine* 49.18 (2016), pp. 716–723. DOI: 10.1016/j.ifacol.2016.10.250.

[68] Milan Korda and Igor Mezić. “Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control.” In: *Automatica* 93 (July 2018), pp. 149–160. DOI: 10.1016/j.automatica.2018.03.046.

[69] E. Kaiser, J. N. Kutz, and S. L. Brunton. *Data-driven discovery of Koopman eigenfunctions for control*. 2017. arXiv: 1707.01146 [math.OC].

[70] Michael Schmidt and Hod Lipson. “Distilling Free-Form Natural Laws from Experimental Data.” In: *Science* 324.5923 (Apr. 2009), pp. 81–85. DOI: 10.1126/science.1165893.

[71] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. Vol. 1. MIT press Cambridge, 2016.

[72] Enoch Yeung, Soumya Kundu, and Nathan Hodas. *Learning Deep Neural Network Representations for Koopman Operators of Nonlinear Dynamical Systems*. 2017. eprint: 1708.06850 (cs.LG).

[73] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. “Learning Koopman Invariant Subspaces for Dynamic Mode Decomposition.” In: *Advances in Neural Information Processing Systems*. 2017, pp. 1130–1140.

[74] Christoph Wehmeyer and Frank Noé. “Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics.” In: *The Journal of Chemical Physics* 148.24 (June 2018), p. 241703. DOI: 10.1063/1.5011399.

[75] Andreas Mardt, Luca Pasquali, Hao Wu, and Frank Noé. “VAMPnets for deep learning of molecular kinetics.” In: *Nat Commun* 9.1 (Jan. 2018). DOI: 10.1038/s41467-017-02388-1.

[76] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. “Deep learning for universal linear embeddings of nonlinear dynamics.” In: *Proc. Natl. Acad. Sci. U.S.A.* (2017). arXiv: 1712.09707 [math.DS]. Submitted.

[77] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. “PDE-Net: Learning PDEs from Data.” In: (Nov. 2017). arXiv: 1710.09668 [math.NA].

[78] Maziar Raissi and George Em Karniadakis. “Hidden physics models: Machine learning of nonlinear partial differential equations.” In: *J. Comput. Phys.* 357 (Mar. 2018), pp. 125–141. DOI: 10.1016/j.jcp.2017.11.039.

[79] Pantelis R Vlachas, Wonmin Byeon, Zhong Y Wan, Themistoklis P Sapsis, and Petros Koumoutsakos. *Data-Driven Forecasting of High-Dimensional Chaotic Systems with Long-Short Term Memory Networks*. 2018. arXiv: 1802.07486 [physics.comp-ph].

[80] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. “Traffic Flow Prediction With Big Data: A Deep Learning Approach.” In: *IEEE Trans. Intell. Transport. Syst.* 16.2 (2014), pp. 1–9. DOI: 10.1109/tits.2014.2345663.

[81] Bicky A. Marquez, Laurent Larger, Maxime Jacquot, Yanne K. Chembo, and Daniel Brunner. “Dynamical complexity and computation in recurrent neural networks beyond their fixed point.” In: *Sci Rep* 8.1 (Feb. 2018). DOI: 10.1038/s41598-018-21624-2.

[82] Kangbeom Cheon, Jaehoon Kim, Moussa Hamadache, and Dongik Lee. “On Replacing PID Controller with Deep Learning Controller for DC Motor System.” In: *JOACE* 3.6 (2015), pp. 452–456. DOI: 10.12720/joace.3.6.452-456.

[83] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. “Human-level control through deep reinforcement learning.” In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. DOI: 10.1038/nature14236.

[84] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. “Occam’s Razor.” In: *Information Processing Letters* 24.6 (Apr. 1987), pp. 377–380. DOI: 10.1016/0020-0190(87)90114-1.

[85] Leo Breiman. “Bagging predictors.” In: *Mach Learn* 24.2 (Aug. 1996), pp. 123–140. DOI: 10.1007/bf00058655.

[86] Robert E. Schapire. “The strength of weak learnability.” In: *Mach Learn* 5.2 (June 1990), pp. 197–227. DOI: 10.1007/bf00116037.

[87] D.H. Wolpert and W.G. Macready. “No free lunch theorems for optimization.” In: *IEEE Trans. Evol. Computat.* 1.1 (Apr. 1997), pp. 67–82. DOI: 10.1109/4235.585893.

[88] D.H. Wolpert and W.G. Macready. “Coevolutionary Free Lunches.” In: *IEEE Trans. Evol. Computat.* 9.6 (Dec. 2005), pp. 721–735. DOI: 10.1109/tevc.2005.856205.

[89] Guido F Smits and Mark Kotanchek. “Pareto-front exploitation in symbolic regression.” In: *Genet. Program. theory Pract. II* (2005), pp. 283–299. DOI: doi:10.1007/0-387-23254-0_17.

[90] Markus Quade. *sparsereg - Collection of Modern Sparse Regression Algorithms*. Version 0.8.5. Feb. 2018. URL: https://github.com/ohjeah/sparsereg. DOI: 10.5281/zenodo.1173754.

[91] G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, and A. Mueller. “Scikit-learn. Machine Learning Without Learning the Machinery.” In: *GetMobile: Mobile Comp. and Comm.* 19.1 (June 2015), pp. 29–33. DOI: 10.1145/2786984.2786995.

[92] Markus Quade, Julien Gout, and Markus Abel. *glyph - Symbolic Regression Tools*. Version 0.3.5. Jan. 2018. URL: https://github.com/Ambrosys/glyph. DOI: 10.5281/zenodo.1156654.

[93] Philipp Oswald. “Experimental investigations of active and passive drag-reducing devices over a D-shaped bluff body.” Master’s Thesis. Institut für Strömungsmechanik, Technische Universität Braunschweig, Sept. 2017.

[94] Leonor Michaelis and Maud Leonora Menten. “Die Kinetik der Invertinwirkung.” In: *Biochem.* 49 (1913), pp. 333–369.

[95] Bryan C. Daniels and Ilya Nemenman. “Efficient Inference of Parsimonious Phenomenological Models of Cellular Dynamics Using S-Systems and Alternating Regression.” In: *PLoS ONE* 10.3 (Mar. 2015), e0119821. URL: https://doi.org/10.1371/journal.pone.0119821.

[96] Andrey Shilnikov, René Gordon, and Igor Belykh. “Polyrhythmic synchronization in bursting networking motifs.” In: *Chaos* 18.3 (Sept. 2008), p. 037120. DOI: 10.1063/1.2959850.

[97] Jeremy Wojcik, Robert Clewley, and Andrey Shilnikov. “Order parameter for bursting polyrhythms in multifunctional central pattern generators.” In: *Phys. Rev. E* 83.5 (May 2011), p. 056209. DOI: 10.1103/physreve.83.056209.

[98] Rico Jonschkowski and Oliver Brock. “Learning state representations with robotic priors.” In: *Auton Robot* 39.3 (July 2015), pp. 407–428. DOI: 10.1007/s10514-015-9459-7.

[99] M.D. Schmidt and H. Lipson. “Coevolution of Fitness Predictors.” In: *IEEE Trans. Evol. Computat.* 12.6 (Dec. 2008), pp. 736–749. DOI: 10.1109/tevc.2008.919006.

[100] Dario Izzo, Francesco Biscani, and Alessio Mereta. “Differentiable Genetic Programming.” In: *European Conference on Genetic Programming*. Springer. 2017, pp. 35–51.

[101] Matt Johnson Dougal Maclaurin David Duvenaud. *autograd*. 2018. URL: https://github.com/HIPS/autograd (visited on 04/29/2018).

[102] Eusebius Doedel et al. *AUTO - Software for Continuation and Bifurcation Problems in Ordinary Differential Equations*. 2011. URL: http://cmvl.cs.concordia.ca/auto/.

[103] Jörn Malich. “Modeling and Prediction of Extended Multiscale Systems Using Gaussian Processes and Symbolic Regression.” Master’s Thesis. Institute for Physics and Astronomy, University of Potsdam, Feb. 2016.

[104] G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, and A. Mueller. “Scikit-learn. Machine Learning Without Learning the Machinery.” In: *GetMobile: Mobile Comp. and Comm.* 19.1 (June 2015), pp. 29–33. DOI: 10.1145/2786984.2786995.

[105] Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, Eurika Kaiser, and J. Nathan Kutz. “Chaos as an intermittently forced linear system.” In: *Nat Commun* 8.1 (May 2017). DOI: 10.1038/s41467-017-00030-8.

[106] *Handbook of Chaos Control*. Wiley-VCH Verlag GmbH & Co. KGaA, Oct. 2007. DOI: 10.1002/9783527622313.

[107] Edward Ott, Celso Grebogi, and James A. Yorke. “Controlling chaos.” In: *Phys. Rev. Lett.* 64.11 (Mar. 1990), pp. 1196–1199. DOI: 10.1103/physrevlett.64.1196.

[108] Nutonian, a DataRobot company. *Eureqa: The A.I.-Powered Modeling Engine*. 2018. URL: https://www.nutonian.com/products/eureqa/ (visited on 04/29/2018).

[109] Evolved Analytics. *Delivering Data-Driven Solutions*. URL: http://evolved-analytics.be/ (visited on 04/30/2018).

[110] Paul Boon, Elien De Cock, Ann Mertens, and Eugen Trinka. “Neurostimulation for drug-resistant epilepsy. a systematic review of clinical evidence for efficacy, safety, contraindications and predictors for response.” In: *Curr. Opin. Neurol.* (Jan. 2018), p. 1. DOI: 10.1097/wco.0000000000000534.

[111] Conor Ryan and Maarten Keijzer. “An analysis of diversity of constants of genetic programming.” In: *European Conference on Genetic Programming*. Springer. 2003, pp. 404–413.

[112] Markus F Brameier and Wolfgang Banzhaf. *Genetic Programming*. Springer Berlin Heidelberg, 2000. DOI: 10.1007/b75085.

[113] Michael O'Neil and Conor Ryan. "Grammatical evolution." In: *Grammatical Evolution*. Springer US, 2003. Chap. Grammatical Evolution, pp. 33–47. DOI: 10.1007/978-1-4615-0447-4_4.

[114] M Schmidt and H Lipson. "Comparison of Tree and Graph Encodings as Function of Problem Complexity." In: *Gecco 2007: Genetic and Evolutionary Computation Conference, Vol 1 and 2* (2007), pp. 1674–1679. DOI: 10.1145/1276958.1277288.

[115] John R Koza. *Genetic Programming II, Automatic Discovery of Reusable Subprograms*. MIT Press, Cambridge, MA, 1992.

[116] Michael Affenzeller, Stephan Winkler, Stefan Wagner, and Andreas Beham. *Genetic Algorithms and Genetic Programming. Modern Concepts and Practical Applications*. Chapman and Hall/CRC, Apr. 2009. DOI: 10.1201/9781420011326.

[117] Edgar Galvan-Lopez. "Efficient graph-based genetic programming representation with multiple outputs." In: *Int. J. Autom. Comput.* 5.1 (Jan. 2008), pp. 81–89. DOI: 10.1007/s11633-008-0081-4.

[118] Julian F Miller. "Cartesian genetic programming." In: *Cartesian Genetic Programming*. Springer, 2011, pp. 17–34.

[119] Julian Francis Miller and Simon L. Harding. "Cartesian genetic programming." In: *Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation - GECCO '08*. Springer. ACM Press, 2008, pp. 121–132. DOI: 10.1145/1388969.1389075.

[120] Julian F. Miller and Simon L. Harding. *Cartesian Genetic Programming Tutorial*. Gecco Companion, Philadelphia, July 8, 2012. (Visited on 04/29/2018).

[121] J.A. Walker and J.F. Miller. "The Automatic Acquisition, Evolution and Reuse of Modules in Cartesian Genetic Programming." In: *IEEE Trans. Evol. Computat.* 12.4 (Aug. 2008), pp. 397–417. DOI: 10.1109/tevc.2007.903549.

[122] James Alfred Walker and Julian Francis Miller. "Evolution and acquisition of modules in cartesian genetic programming." In: *European Conference on Genetic Programming*. Springer. 2004, pp. 187–197.

[123] T. Perkis. "Stack-based genetic programming." In: *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*. IEEE. IEEE, 1994, pp. 148–153. DOI: 10.1109/icec.1994.350025.

[124] Lee Spector and Alan Robinson. "Genetic programming and autoconstructive evolution with the push programming language." In: *Genetic Programming and Evolvable Machines* 3.1 (2002), pp. 7–40.

[125] Maarten Keijzer. "Push-forth. a light-weight, strongly-typed, stack-based genetic programming language." In: *Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion - GECCO '13 Companion*. ACM Press, 2013, p. 1635. DOI: 10.1145/2464576.2482742.

[126] Lee Spector and Thomas Helmuth. "Effective simplification of evolved push programs using a simple, stochastic hill-climber." In: *Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion - GECCO Comp '14*. ACM. ACM Press, 2014, pp. 147–148. DOI: 10.1145/2598394.2598414.

[127] Lee Spector, Brian Martin, Kyle Harrington, and Thomas Helmuth. "Tag-based modules in genetic programming." In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11*. ACM. ACM Press, 2011, pp. 1419–1426. DOI: 10.1145/2001576.2001767.

[128] William La Cava, Kourosh Danai, and Lee Spector. "Inference of compact nonlinear dynamic models by epigenetic local search." In: *Eng. Appl. Artif. Intell.* 55 (Oct. 2016), pp. 292–306. DOI: 10.1016/j.engappai.2016.07.004.

[129] Christian Veenhuis. "Structure-Based Constants in Genetic Programming." In: vol. 8154. Lecture Notes in Computer Science September 2013. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. DOI: 10.1007/978-3-642-40669-0.

[130] Thomas Fernandez and Matthew Evett. "Numeric mutation as an improvement to symbolic regression in genetic programming." In: *International Conference on Evolutionary Programming*. Springer. 1998, pp. 251–260.

[131] L.M. Howard and D.J. D'Angelo. "The GA-P: A genetic algorithm and genetic programming hybrid." In: *IEEE Expert* 10.3 (June 1995), pp. 11–15. DOI: 10.1109/64.393137.

[132] François-Michel De Rainville, Félix-Antoine Fortin, Marc-André Gardner, Marc Parizeau, and Christian Gagné. "Deap. enabling nimbler evolutions." In: *SIGEVOlution* 6.2 (Feb. 2014), pp. 17–26. DOI: 10.1145/2597453.2597455.

[133] Markus Quade, Markus Abel, J. Nathan Kutz, and Steven L. Brunton. "Sparse identification of nonlinear dynamics for rapid model recovery." In: *Chaos* 28.6 (28 June 2018), p. 063116. DOI: 10.1063/1.5027470.

[134] J. Friedman, T. Hastie, and R. Tibshirani. *A note on the group lasso and a sparse group lasso*. 2010. arXiv: 1001.0736 [math.ST].

[135] Jie Peng, Ji Zhu, Anna Bergamaschi, Wonshik Han, Dong-Young Noh, Jonathan R. Pollack, and Pei Wang. "Regularized multivariate regression for identifying master predictors with application to integrative genomics study of breast cancer." In: *Ann. Appl. Stat.* 4.1 (Mar. 2010), pp. 53–77. DOI: 10.1214/09-aoas271.

[136] Benjamin Strom, Steven L. Brunton, and Brian Polagye. "Intracycle angular velocity control of cross-flow turbines." In: *Nat. Energy* 2.8 (June 2017), p. 17103. DOI: 10.1038/nenergy.2017.103.

[137] Russell R. Barton and John S. Ivey. "Nelder-Mead Simplex Modifications for Simulation Optimization." In: *Manage. Sci.* 42.7 (July 1996), pp. 954–973. DOI: 10.1287/mnsc.42.7.954.

[138] Ruiying Li, Bernd R. Noack, Laurent Cordier, Jacques Borée, and Fabien Harambat. "Drag reduction of a car model by linear genetic programming control." In: *Exp Fluids* 58.8 (July 2017). DOI: 10.1007/s00348-017-2382-2.

A GP REPRESENTATION

Symbolic regression is a combined problem of [53]

- structure discovery,
- variable/feature selection and
- constant/parameter optimization.

While structure discovery and feature selection by the genetic algorithm using a common data structure, *cf.* Section A.1, determining optimal parameter values remains a challenge [111] and has to be dealt with separately. Common approaches for dealing with parameters are listed in Section A.2.

A.1 STRUCTURE REPRESENTATIONS

This section gives an overview about different structure presentations of expressions in the context of genetic programming based symbolic regression. GP has a lot of heuristics and comes in many flavors. Besides the listed ones, there is also Linear Genetic Programming [112], Grammatical Evolution [113] and many more. The *genotype* of an expression is the expression encoded in its representation. The *phenotype* of an expression is the expression itself up to mathematical equivalence.

A.1.1 *Tree Based Representation*

For a detailed description see also Section 3.2.1 and Section 4.2.2.
The simplest way to represent a program is an expression tree, *cf.* Fig. A.1. It is used in the early implementations of genetic programming based symbolic regression [34]. The nodes of the tree represent program primitives. The edges represent how the primitives are connected. For symbolic regression, leaf nodes represent variables or constants and all other nodes represent functions. Typically, the genotype translates directly into the phenotype, *i. e.* there are no non-coding genes and the encoding is not compressed. Trees are generated by growing them from top to bottom, either to a fixed length or with a probabilistic stopping criteria. Both main methods of tree manipulation – mutation and crossover – are based on generating or exchanging subtrees. This makes the tree based encoding susceptible to bloat. Variable program size allows for simplification. Similarly to mutation, a subtree is replaced by a mathematically equivalent, but smaller tree. Trees are the most intuitive representation, however, they also lack in efficiency (no compression) or are unsuitable for convergence speed improving heuristics.

Multiple Outputs

In the naive approach to multiple output functions are represented by a list of single output/ one dimensional functions. The implementation of the genetic op-

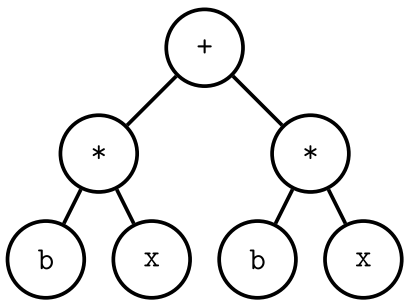


Figure A.1: The expression $bx + bx$ encoded as a tree. The encoding needs 7 nodes and has 4 unique symbols. The depth of the tree is 3.

erators is straight forward and wraps around the corresponding single output genetic operators:

- **Creation**: Vectorized version of the single dimension tree creation method.
- **Mutation**: Pick one (or many) of the n subtrees and apply the single output mutation operator.
- **Crossover**: Pick two out of the $2n$ subtrees and apply the single output crossover operator. Return a new offspring-based multi-output tree.

The information shared between the different dimensions is minimal. Common subexpressions, e. g. coupling terms, have to be represented/discovered multiple times. Alternatively, one can use a single tree with multiple entry points, *cf.*

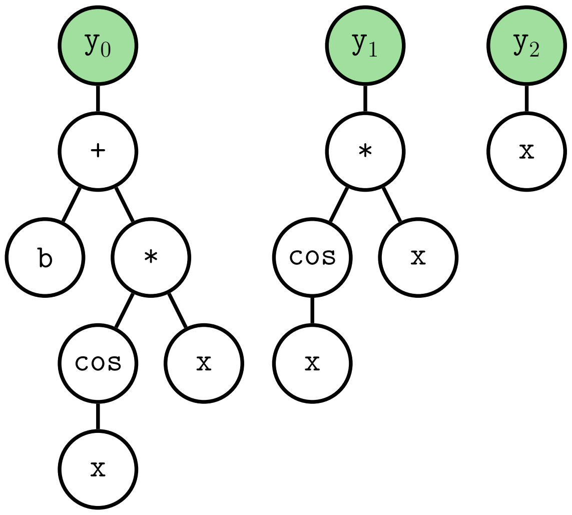


Figure A.2: Tree list representation of $\vec{y} = \vec{f}(x)$ with $y_0 = x\cos(x) + b$, $y_1 = x\cos(x)x$ and $y_2 = x$. A total of 14 nodes are used to represent the function.

Fig. A.2 and Fig. A.3. However, this constraints possible variations of the tree since entrypoints have to remain valid.

A.1.2 *Graph-based Representation*

An expression often has repeating parts, e. g. the argument when approximating a function by a Taylor series or coupling terms when identifying a dynamical system.

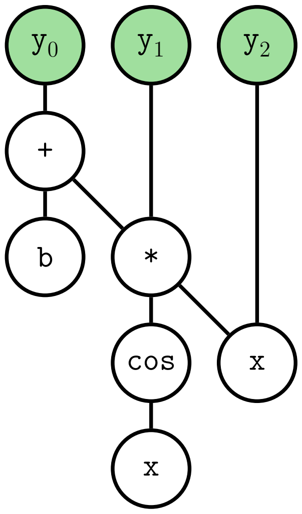


Figure A.3: Single tree multiple entrypoints representation of $\vec{y} = \vec{f}(x)$ with $y_0 = x\cos(x) + b$, $y_1 = x\cos(x)x$ and $y_2 = x$. A total of 9 nodes are used to represent the function.

With tree based genetic programming the repeating parts have to be discovered and stored multiple times within the same tree. Although is has been shown that this does not necessarily slow down the optimization [114]. Graphs are the natural extension to trees, allowing child nodes to be shared between (grand-) parents. E. g. encoding the expression $bx + bx$ in a graph, *cf.* Fig. A.4, requires three nodes less than using a tree, *cf.* Fig. A.1.

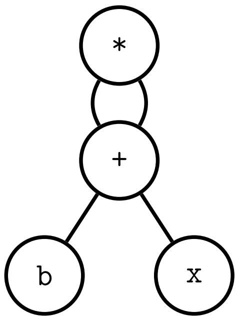


Figure A.4: The expression $bx + bx$ encoded as a graph. The encoding needs 4 nodes and has 4 unique symbols.

Automatic Defined Functions and Modularity

Automatic Defined Functions (ADF) [115] are an extension to tree-based GP. A primitive node of the main tree is allowed be represented by another tree. If there are n unique ADFs in the primitive set, the genotype of the individual is represented by $n+1$ trees. ADFs describe a subset of all possible graphs, *cf.* Fig. A.5. The main idea of ADFs is that the program is organized in modules enabling parametrized reuse and hierarchical invocation [116]. Similarly, pointer primitives [117] allow for program modularization.

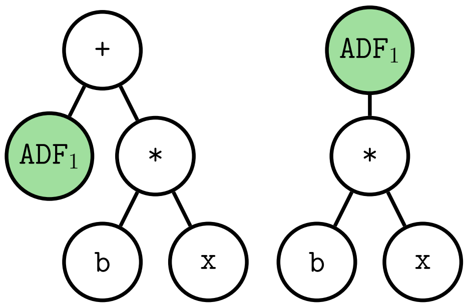


Figure A.5: The expression $bx + bx$ encoded as tree with one ADF. The encoding needs 9 nodes and has 5 unique symbols.

Cartesian Genetic Programming

Typically, Cartesian Genetic Programming (CGP) [117, 118, 119] is used instead of working indirectly encoding graphs. In CGP, programs are represented by a two dimensional grid of program primitives. The genotype is a list of integers representing the primitives and their connections [120]. The genes are either links to data or links to function lookup table. Connections between genes are constrained; genes may only connect to a gene between one and n_{back} columns earlier in the program. Output genes may connect to any other gene. This ensures that the grid represents a directly acyclic graph with additional non-coding genes, i.e. genes which are not connected to any output gene and therefore do not contribute to the phenotype of the program. Fig. A.6 shows the expression $bx + bx$ encoded in a CGP program with one row and five columns. The first number of the gene links to the function lookup table, the following numbers connect to other genes. While output genes only connect to other genes, input genes can only be connected to. The non-coding genes in the program are genes number three, four and five. Programs

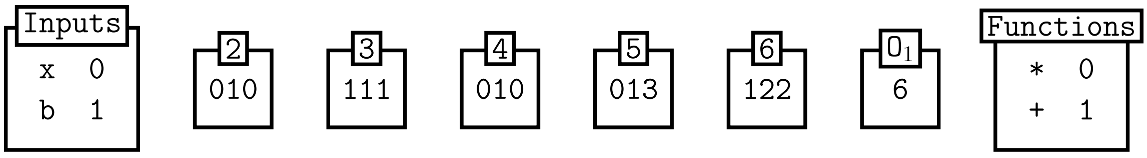


Figure A.6: The expression $bx + bx$ encoded as a Cartesian register. The program size is fixed to 2 input genes, 1 row, 5 columns and 1 output gene. The representation has *non-coding* genes (3, 4, 5).

are changed with point mutations. A point mutation changes either changes the function or connection link of a gene. It is recommenced not to use crossover [118]. Due to the fixed maximum program size and the way connections are allowed, CGP programs are inherently sparse [119] and do not suffer the same problems with bloat as its tree based counterpart. Representing another dimension is as simple as adding another output gene without further constraints on possible program manipulations. Therefore, CGP lends itself for system identification [114]. Similarly to ADF, CGP has evolvable and reusable submodules [121, 122].

A.1.3 *Stack-based Representation*

In stack-based GP, programs are expressed in a stack based programming language [123, 124, 125]. The stack genotype uses first-in-last-out push and pull actions. A program in postfix notation is parsed as follows: iteratively each primitive of the program is considered. If it has zero arity (variable, parameter) it is pushed to the stack. If it is a function and the stack is greater or equal than the functions arity, a new zero arity expression constructed by pulling arguments for the function from the stack. The constructed expression is pushed to the stack. If however, the stack cannot provide enough arguments for the function, it is simply ignored. Finally, after each primitive of the program has been parsed, the resulting expression is the first pulled from the stack (there may be more than one). An example is given in Fig. A.7. Due to the simplicity of the data structure and the fact that invalid combinations of primitives are non-coding , stack-based GP is much less constrained compared to tree-based GP and allows for automatic simplification [126], ADFs [116] module/motif discovery [127] or epigenetic local search [128].

```
program:  [x, -, x, b, +, b, x, +, *]
```

	instruction	stack
(x)	push x	[x]
(-)	ignore	[x]
(x)	push x	[x, x]
(b)	push b	[x, x, b]
(+)	pull b, pull x push b+x	[x, b+x]
(b)	push b	[x, b+x, b]
(x)	push x	[x, b+x, b, x]
(+)	pull b, pull x push b+x	[x, b+x, b+x]
(*)	pull b+x, pull b+x push (b+x)*(b+x)	[x, (b+x)*(b+x)]

Figure A.7: The expression $bx + bx$ encoded as postfix stack. The encoding needs 9 nodes and has 5 unique symbols.

A.2 constants in genetic programming

A.2.1 *Ephemeral Random Constants*

Ephemeral Random Constantss (ERCs) [34] are a special leaf node holding a random numeric value drawn from a distribution, e. g. $\mathcal{U}_{[-1,1]}$. If a new candidate solution is created, either by creating a new one or by breeding old ones, all ERC values are drawn independently.

Since ERCs do not require any changes to the genetic algorithm or special operator to alter their values, they are relatively easy to implement and do not change the computational complexity of the optimization algorithm. However, in this case, constant optimization – like structure discovery – is fitness guided and there is no mechanism for a more systematic search.

A.2.2 *Structure-based Constants*

Structure-based constants [129] introduce a new primitive function which maps the structural properties of its child tree to real numbers, e. g. $SC(T_1, T_2) = \frac{\|T_1\|}{\|T_2\|}$, where $\|T_{1/2}\|$ is the number of nodes in the left or right subtree correspondingly. Other properties to be considered are the depth of the subtree or the maximum artiy of any node in the subtree. Usually, structure-based constants are scaled to an interval $[c_{\min}, c_{\max}]$.

Structure-based constants are optimized – like ERCs – by the genetic algorithm and therefore they do not change the computational complexity. They only require a few extra parameters and are typically even easier to implement compared to ERCs as most genetic programming frameworks allow for customization of the primitives.

A.2.3 *Symbolic Constants*

Symbolic constants introduce a more systematic search strategy. This is achievable by either introducing a special mutation operator or by formulating a layered optimization problem. Using a mutation operator, the value of random constants is changed by adding a random increment instead of replacing it altogether [130], *i. e.* $c \mapsto c + \xi$ and $\xi \sim \mathcal{N}(0, 1)$, where c is the value of the constant and ξ is a normal distributed random variable.

In a layered optimization problem for each discovered structure a the subproblem $\mathbf{c}^* = \operatorname{argmin}_{\mathbf{c}} \mathcal{L}$ has to be solved, where $\mathbf{c}$ is the vector of constants and $\mathcal{L}$ is the objective function, e. g. the normalized root mean squared error. See also Section 3.3 for more details.

Optionally, one can use evolutionary optimization, e. g. genetic algorithms [131], or gradient based optimization [53, 41, 54]. Symbolic regression has the advantage of symbolic differentiability, which will speed up gradient based method.

However, formulating a layered optimization problem increases the computational complexity multiplicatively and not for every structure it is worthwhile to determine the optimal constant values as they might still be comparatively bad. Therefore, layered optimization often is accompanied by additional heuristics, *cf.* Appendix C, lowering the computational complexity. This can include running only a small number of iterations or only optimizing the constants of candidate solutions with promising structures.

B SOFTWARE CONTRIBUTIONS

B.1 GLYPH

`Glyph` is a python 3 library based on `DEAP`[132] providing abstraction layers for symbolic regression problems.

It comes with batteries included:

- predefined primitive sets,
- n-dimensional expression tree class,
- symbolic and structural constants,
- interfacing constant optimization to `scipy.optimize`,
- easy integration with `joblib` or `dask.distributed`,
- symbolic constraints,
- boilerplate code for logging, check-pointing, break conditions and command line applications,
- rich set of algorithms.

`Glyph` also includes a plug and play command line application `glyph-remote` which lets non-domain experts apply symbolic regression to their optimization tasks.

List of contributors

Core contributors (prior to open source):

- Markus Quade
- Julien Gout

Open source contributors can be found at `https://github.com/Ambrosys/glyph/graphs/contributors`.

Archive: Zenodo

NAME: Ambrosys/glyph
PERSISTENT IDENTIFIER: `http://doi.org/10.5281/zenodo.801819`
LICENCE: LGPL
PUBLISHER: Markus Quade
VERSION PUBLISHED: 0.3.5
DATE PUBLISHED: 22.01.18

Code repository: Github

NAME: glyph
PERSISTENT IDENTIFIER: `https://github.com/Ambrosys/glyph`
LICENCE: LGPL
DATE PUBLISHED: 08.12.16

B.2 CARTESIAN

`Cartesian` is a small library implementing cartesian genetic programming [118]. The library is implemented in Python and adheres to the well known `scikit-learn` interface[91].

The basic components are provided:

- data structure
- $1 + \lambda$ algorithm
- symbolic, ephemeral and structured constants

`Cartesian` can provide insight into which constant type you should use for your problem as is naturally distinguishes between the total number of function evaluations and iteration of the genetic algorithm.

Archive: Zenodo

NAME: Ohjeah/cartesian
PERSISTENT IDENTIFIER: `https://doi.org/10.5281/zenodo.1202180`
LICENCE: LGPL
PUBLISHER: Markus Quade
VERSION PUBLISHED: 0.1.4
DATE PUBLISHED: 19.03.18

Code repository: Github

NAME: cartesian
PERSISTENT IDENTIFIER: `https://github.com/Ohjeah/cartesian`
LICENCE: LGPL
DATE PUBLISHED: 21.07.17

B.2.1 *Example Program*

```python
from sklearn.datasets import make_regression
from sklearn.utils.validation import check_random_state
from sklearn.model_selection import train_test_split
from cartesian import Symbolic

rng = check_random_state(1337)

```

```
x, y, coef = make_regression(
    n_features=2, n_informative=1, n_targets=1, random_state=rng, coef=True)
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=rng)

est = Symbolic(
    n_const=2,
    random_state=rng,
    n_columns=5,
    n_rows=2,
    max_nfev=100000,
    n_jobs=-1,
    f_tol=1e-4)
est.fit(x_train, y_train)
print(est.res)
print(est.score(x_test, y_test))
```

B.3 sparsereg

`sparsereg` is a collection of modern sparse (regularized) regression algorithms. The library is implemented in Python and adheres to the well known `scikit-learn` interface[91]. It implements (variations) of the following algorithms:

- Fast Function Extraction `FFX` [23],
- Evolutionary Feature Synthesis `EFS` [51],
- Bootstrapped Adaptive Threshold Selection `BOATS` [30],
- Sparse Identification of Nonlinear Dynamics `SINDy` [29], as well as its variants/extensions `Abrupt-SINDy` [133] and Group Sparse Lasso based SINDy [134, 135].

Archive: Zenodo

NAME: Ohjeah/sparsereg
PERSISTENT IDENTIFIER: `https://doi.org/10.5281/zenodo.1182579`
LICENCE: LGPL
PUBLISHER: Markus Quade
VERSION PUBLISHED: 0.8.4
DATE PUBLISHED: 21.02.18

Code repository: Github

NAME: sparsereg
PERSISTENT IDENTIFIER: `https://github.com/Ohjeah/sparsereg`
LICENCE: LGPL
DATE PUBLISHED: 30.01.17

B.3.1 *Example Program*

The example highlights system identification of a simple harmonic oscillator using `SINDy`.

```python
import warnings

import numpy as np
from scipy.integrate import odeint
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.utils import check_random_state

from sparsereg.model import SINDy


def rhs_harmonic_oscillator(y, t):
    dy0 = y[1]
    dy1 = -0.3 * y[0]
    return [dy0, dy1]


x0 = [0, 1]
t = np.linspace(0, 10, 1000)
x = odeint(rhs_harmonic_oscillator, x0, t)

x_train, x_test = x[:750], x[750:]

kw = dict(fit_intercept=True, normalize=False)
model = SINDy(dt=t[1] - t[0], degree=2, alpha=0.3, kw=kw)

rng = check_random_state(42)
cv = KFold(n_splits=5, random_state=rng, shuffle=False)
params = {"alpha": [0.1, 0.2, 0.3, 0.4, 0.5], "threshold": [0.1, 0.3, 0.5]}
grid = GridSearchCV(model, params, cv=cv)

with warnings.catch_warnings():  # suppress matrix ill-conditioned warning
    warnings.filterwarnings("ignore")
    grid.fit(x_train)
selected_model = grid.best_estimator_

print("Score on test data ", selected_model.score(x_test))
print(
    "Selected hyperparameter (alpha, threshold): ",
    selected_model.alpha,
    selected_model.threshold,
)
for i, eq in enumerate(selected_model.equations()):
    print("dx_{} / dt = ".format(i), eq)
print(
    "Complexity of the model (sum of coefficients and \
intercetps bigger than the threshold): ",
    selected_model.complexity,
)
```

C CROSS-FLOW TURBINE EXPERIMENT SEATTLE

by Markus Quade[7], Benjamin W. Strom[8]

[7] Universität Potsdam, Institut für Physik und Astronomie, Karl-Liebknecht-Straße 24/25, 14476 Potsdam, Germany
[8] Department of Mechanical Engineering, University of Washington, Box 352600, NE Stevens Way, Seattle, Washington 98195, USA

This chapter is based on an internal report. We implement and demonstrate the use of the genetic symbolic regression software `glyph`*, cf. Chapter 5, in a real world experiment setting. Additionally, we optimize the control of a cross-flow turbine using a model-free approach.*

Due to their complex hydrodynamics, accurate low-order analytical models for the instantaneous or average power production of cross-flow turbines do not exist. This is a mainly a result the cyclically varying flow conditions experienced by the blades, which includes full separation, interaction with the wake of upstream blades, and operation in a curvilinear flow field. Control strategies employing machine-learning have previously been shown result in a 58% increase in power output in scale-model cross-flow turbine [136]. By varying the rotation rate of the turbine as a function of azimuthal blade position, the fluid-structure interaction was optimized to increase increase beneficial forcing. Additionally, extrema of torque and angular velocity were aligned, resulting in larger power output during favorable portions of the blade rotation and reducing losses during unfavorable portions.

C.1 PRIOR WORK

The conversion efficiency, also known as power coefficient is defined as

$$c_P = \frac{\tau\omega}{\frac{1}{2}\rho U_\infty^3 A}, \tag{C.1}$$

where τ is the torque passed onto the turbine by the fluid, ω is the rotation frequency, ρ is the fluid density, U_∞ the free stream velocity, and A is the projected area normal to the free stream direction. The rotor rotates according to

$$I\dot{\omega} = \tau_{\text{fluid}} + \tau_{\text{external}}, \tag{C.2}$$

where τ_{fluid} is the torque impaired by the fluid; τ_{external} subsumes losses and control. The torque produced by the rotor and used to calculate the conversion efficiency in Equation C.1 is

$$\tau := \tau'_{\text{fluid}} - I\dot{\omega}. \tag{C.3}$$

Four types of control have been applied: constant torque τ, constant rotation frequency ω, as well as two parametrized rotation frequency profiles:

$$\omega(\theta) = A_0 + A_1 \sin(N\theta + \phi_1) \tag{C.4}$$

and

$$\omega(\theta) = A_0 + A_1 \sin(N\theta + \phi_1) + A_2 \sin(2N\theta + \phi_2) + A_3 \sin(3N\theta + \phi_4), \tag{C.5}$$

where N is the number of turbine blades. The Nelder-Mead downhill simplex optimizer was used to determine the constants A and ϕ, and $N = 2$ was employed. A comprehensive description of the method can be found in [136].

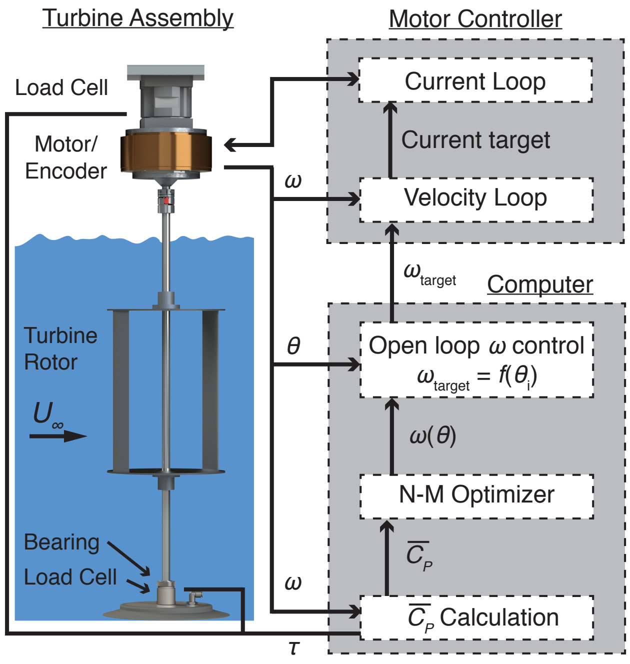


Figure C.1: Taken from [136] with permission. Turbine set-up and control system.

C.2 SETUP FOR glyph

As a proof-of-concept of `Glyph` software for learning control laws for cross-flow turbines, we attempt to recreate or exceed the performance of the above prior study, but no prior assumption of the control law form. To implement this, `Glyph` is restricted to single input (the turbine azimuthal position, θ), single output (the turbine angular velocity, ω) functions that are periodic over a half rotation rate. Like the above functions, the periodicity ensures each of the two blades undergoes identical kinematics.

Table C.1 lists all parameters regarding the use of `Glyph` for the crossflow turbine experiment.

With the parameters, we expect to be able to learn (at a minimum) the optimal sinusoidal waveform found in the previous work, but in the form

$$\omega(\theta) = c_0 + c_1 \sin(2\theta) + c_2 \cos(2\theta) \tag{C.6}$$

Table C.1: GP Parameters for the crossflow turbine experiment

population size	10
max. generations	5
MOO algorithm	NSGA-II
tree generation	*halfandhalf*
min. height	1
max. height	4
selection	*selTournament*
tournament size	2
breeding	*varOr*
recombination	*cxOnePoint*
crossover probability	0.5
crossover max. height	20
mutation	*mutUniform*
mutation probability	0.2
mutation max. height	20
Variables	$\sin(2\theta)$, $\cos(2\theta)$
Operators	$\{+,-\}$
constant type	symbolic
constant number	3
constant optimization	*Nelder-Mead Heuristic*

where the constants are related to Equation C.4 by

$$c_0 = A_0, \quad c_1 = A_1 \cos(\phi_1), \ \ and \ \ c_2 = A_2 \sin(\phi_1). \tag{C.7}$$

Besides raw function evaluations, the largest factor in determining experiment run-time is the need to compile the turbine control algorithm onto a real-time, computer-in-the-loop target (`Simulink Realtime Desktop`). Two steps were used to reduce compile time. First, all control laws for a single generation, with constants left in symbolic form, were ported to `MATLAB` and written to the control software. Second, the control software was programmed to allow changes to the equation constant values during run-time. This allowed for a single compile of the turbine control software per generation.

For each control law, the constants were optimized using the Nelder-Mead downhill simplex algorithm with the RS + S9 improvements for stochastic objective functions [137]. The constant optimization procedure was modified to halt optimization after N_c function evaluations if the control law does not show promise.

C.2.1 *Constant Optimization Heuristic*

1. Determine which constants are present in the control law, so only these are optimized. If no constants are present, evaluate the function.
2. Initialize the simplex. One vertex is chosen at random between -10 and 10. The other vertices are chosen to create a simplex with all side lengths of 5. The direction from the initial vertex to the subsequent ones is random.
3. For N_c function evaluations, use the Nelder-Mead algorithm to walk the simplex towards an optimal solution (see the function evaluation procedure below).
4. Check whether to continue optimizing or not. Optimization continues if a) the best solution found so far for this control law is within 80% of the best solution found for any control law or b) the slope of the optimization curve forecasts performance better than the optimal after N_t more iterations. This slope is calculated by taking the best performance of the simplex after N_c iterations minus the best performance of the initial simplex divided by N_c.
5. If constant optimization is to terminate, return the best performance. Otherwise, continue optimizing for N_t iterations.

C.2.2 *Fitness, Pre-testing and Constraints*

Pre-testing, *i. e.* simulation or heuristic based rejection of control law candidates, is suggested to up the overall experiment as unnecessary evaluation is avoided [138]. However, if the search space is too restricted, the measure of valid candidate solutions might tend to zero and a fitness guided search might not be possible. Therefore, we assign the fitness as follows:

1. Simulate the motion of the turbine under the control law. The simulation is performed using the Runge-Kutta 4th order solver to find $\theta(t)$ from the equation for $\omega(\theta)$.
2. Using the simulation output, determine if the control law is appropriate to test on the experimental turbine. The criteria for an appropriate control law are
   - Minimum velocity greater than zero (Turbine must not rotate backwards)
   - Maximum velocity less than 30 rad/s (Maximum turbine speed)
   - Mean velocity greater than 0.65 rad/s (The turbine must rotate at least once in ten seconds)
   - Maximum angular acceleration induced torque of less than 10 N-m (The limit of the torque cell)

   If the control law and constant set is not appropriate, a fitness is calculated based on the extent to which the control exceeds the criteria. The fitness value returned will always be greater than one. If it is appropriate, proceed to the next step.

3. Implement the control law and constant set on the experimental turbine for a period of 10 seconds while recording data. Calculate the turbine mechanical efficiency over an integer number of turbine rotations. The negative of this values is returned as the fitness.

C.3 DISCUSSION

In this section we discuss the results from our initial test run. Table C.2 lists and compares the results of [136] with the results obtained from an initial run with `Glyph`. We can make two main observations:

- It is easy to find a feedback control which performs better than a open loop control.
- Evaluation of only about 50 candidate solutions is too little to achieve similar performance to a control law based on expert knowledge. Considering the stochastic nature of evolutionary optimization, we can not make any guarantees about reproducibility of the results found or convergence rate. Benefits of evolutionary optimization over random search accumulate slowly, cf. the studies presented in [15, 54].

Control scheme	$\bar{c}_P$	Gain
Constant τ	0.199	
Constant ω	0.203	0%
Equation C.4	0.311	53%
Equation C.5	0.321	59%
`Glyph`	0.260	31 %

Table C.2: Comparison of different control schemes.

We suggest the following improvements:

- Currently, we treat constraints in two places: `Glyph` replaces invalid (zero, constant, infinite) expression with random new ones and the pre-testing code assigns a bad fitness if specific criteria are not met. Maybe we should do both in one place and test in a simulation.
- We should use the semi-analytic solution as a starting point.
- Maybe we can find a transformation, which eliminates the need for constraints.
- Symbolic constants may be replaced by structural or random constants avoiding constant optimization which may be added as a refinement step later on. The possibility of constants optimization during a single experiment should be explored.
- Reinforcement learning variant: Given a current control law, we can try to make small modifications and test its efficiency. Like in simulated annealing,

changes are accepted if either the efficiency is higher of with $p \approx \exp(-\beta \Delta E)$ with ΔE the efficiency difference and β a generalized temperature. β could scale with the confidence in the current control law, e. g. if the control law performs well (lately) and we also have a lot of data (low variance) confidence in the performance is high. If you cannot find a control law with high confidence, revert back to the latest most confident one. This should work better in an actual use case of the turbines. You could chose the hyper parameters such that the expected overall efficiency is maximized. This would need to be programmed closely to the hardware (asynchronous compilation at least).

A further application, which has the potential to be even more transformative for cross-flow turbine performance, will be to include additional state measurements as inputs to the control law. We plan to include one or multiple measures of a turbulent upstream flow. If successful, this will result in control laws that optimize performance in turbulent flows. The turbulence may be endemic to a deployment site, but a more likely scenario is interaction with wakes of upstream turbines in an array. Additional inputs may also be considered, such as turbine thrust and lateral loads. These may be used to optimize the interaction with coherent structures generated during dynamic stall events by sensing their presence. Alternatively, structural loading may be included in the objective function, such that undesirable loading is avoided.