Prediction of dynamical systems by symbolic regression

Markus Quade* and Markus Abel
Universität Potsdam, Institut für Physik und Astronomie, Karl-Liebknecht-Straße 24/25, 14476 Potsdam, Germany and Ambrosys GmbH, David-Gilly-Straße 1, 14469 Potsdam, Germany

Kamran Shafi and Robert K. Niven
School of Engineering and Information Technology, University of New South Wales, Canberra ACT 2600, Australia

Bernd R. Noack
Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur LIMSI-CNRS, BP 133, 91403 Orsay cedex, France and Institut für Strömungsmechanik, Technische Universität Braunschweig, Hermann-Blenk-Straße 37, 38108 Braunschweig, Germany
(Received 15 February 2016; revised manuscript received 13 May 2016; published 13 July 2016)

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.

DOI: 10.1103/PhysRevE.94.012214

I. 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 [1], the prediction of the motion of the planets [2], or the estimation of quantum evolution [3]. The common ingredient of such systems—at least in the 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 modeling languages. Artificial neural networks (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 ANN-type approaches is the difficult-to-interpret black-box nature of the learned models. Symbolic-regression-based approaches, such as genetic programming (GP), provide alternative ML methods that have recently been 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 learned models and a flexible and weakly typed [4] modeling 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. 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 a large number of solutions with low fitness.

*Corresponding author: mquade@uni-potsdam.de

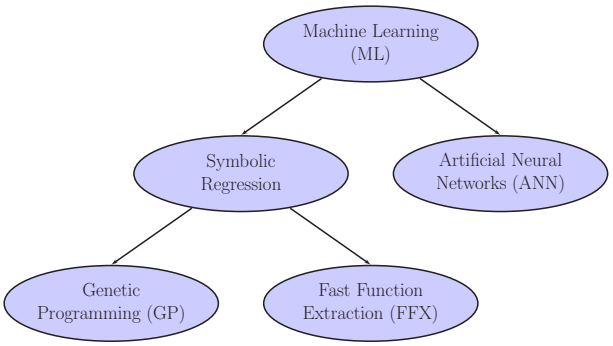


FIG. 1. Illustration of the relation of the methods mentioned in the introduction. We only sketch a part of machine learning, which is relevant for our work.

Symbolic regression subsumes linear regression, generalized linear regression, and generalized additive models into a larger class of methods. Such methods have been used in the past with success to infer equations of dynamical systems directly from data [5–9]. One problem with deterministic chaotic systems, known for a long time, is the sampling of phase space using embedding [10,11]. 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, have been discussed in [12] with numerical methods to obtain suitable differential variables from data. Modern methods such as diffusion maps [13,14] or local linear embedding [15], 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 [16–19]; more recently it has been used for the control of turbulent flow systems [20,21]. In these past publications it is demonstrated how to find symbolic equations in a very general form.

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 package (FFX) [22]. A more general but usually slower method is implemented as a genetic programming algorithm (GP) based on the DEAP package [23]. Subsequent simplification is obtained using the SYMPY package [24]. All mentioned "packages" are Python software packages; of course, any other programming framework with similar functionality will do as well.

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 behavior 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 spatiotemporal 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 [25]. 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 [26]. 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 Sec. II we discuss the methods followed by details of our implementation in Sec. III. This section is followed by the longer Sec. IV, where results are presented for the above-mentioned example systems. We end the paper with a summary and conclusions, Sec. V.

II. 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 nonlinear functions. For deterministic systems, with the advent of modern computers, quantities such as fractal dimensions, Lyapunov exponents, and entropies can also be computed to make systems comparable in dynamics [27,28]. These analyses further allow the rough characterization of the type and number of orbits of a dynamical system [29]. On the other hand, embedding techniques have been developed to reconstruct the dynamics of a high-dimensional system from lower-dimensional time series [10,11,30].

These techniques have a number of limitations with respect to accuracy [31] 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 [31] of the data, since chaos "destroys" information. This can be seen very

clearly by the shift map example [28]. However a system on a regular orbit, even marked with complicated equations, might be predicted accurately. For high-dimensional systems, one needs a large number of data to overcome the "curse of dimensionality" [27]. In fact it can be shown that for each dimension, the number of data needed increases on a power-law basis [27,32]. Eventually, the direct inference of the underlying equations of motion from data can be approached using regression methods, such as Kalman filtering, general linear models (GLMs), generalized additive models (GAMs), or even more general schemes; see [33] and references therein. Apart from the equations themselves, partial derivatives often have to be estimated [12], 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 [13,15,16,34]. 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.

A. 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 such as GLMs or GAMs are contained in such a description. A recent implementation of GLMs is fast function extraction (FFX) [22], which is explained briefly in Sec. II A 2. Genetic programming, explained in Sec. II A 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.

1. Genetic programming

Genetic programming 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 [35]. The algorithm seeks an optimal algorithm, in our case a function, using evolutionary, or "genetic," strategies. The pioneering work was established by [36]. We can briefly describe it as follows: in GP we can represent formulas as expression trees, such as that shown in Fig. 2. Nonterminal nodes are filled with elements from a basic function set defined

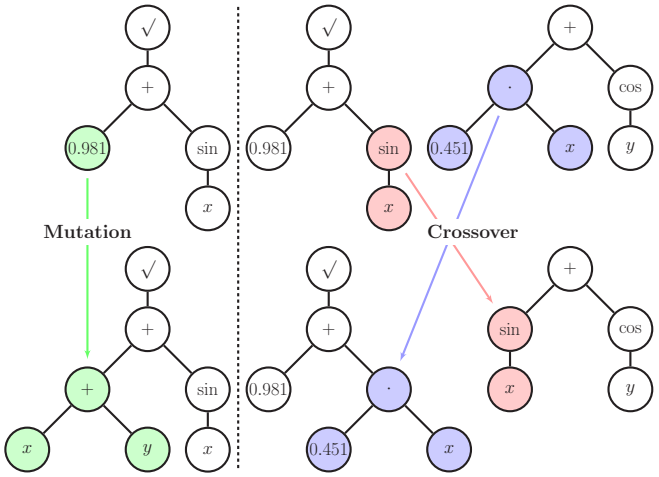


FIG. 2. Illustration of the genetic programming mutation and crossover. The upper left expression tree describes the function $f(x,y) = \sqrt{0.981 + \sin(x)}$. Mutation is conducted by picking a random subtree, here the single terminal node 0.981, and replacing it with a new random expression tree. Similarly, the crossover operator (right) takes two expression trees and swaps two random subtrees.

by the meta-rules. Terminal nodes consist of variables or parameters. Given the optimization problem

$$f^* = \underset{f}{\operatorname{argopt}}\, \Gamma, \tag{1}$$

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:

(1) *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.

(2) *evaluate*. The offspring O_t are evaluated; i.e., their fitness is calculated.

(3) *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. 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 favorable 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 such as GLMs or linear programming [34].

Algorithm 1 Top level description of a GP algorithm

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

2. FFX and the elastic net

Here we briefly summarize the FFX algorithm of McConaghy *et al.* [22]. This is a symbolic regression algorithm based on a combined generalized linear model and elastic net approach:

$$f(\vec{x}) = a_0 + \sum_{i=1}^{N_B} a_i \phi_i(\vec{x}), \tag{2}$$

where $\{a_i\}$ are a set of coefficients to be determined, and $\{\phi_i\}$ are an overdetermined set of basis functions described by a heuristic, simplicity-driven set of rules (e.g., highest allowed polynomial exponent, products, nonlinear 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 [37]

$$\vec{a}^* = \underset{\vec{a}}{\operatorname{argmin}} ||y - f(\vec{x},\vec{a})||_2 + \eta\rho||\vec{a}||_1 + (1-\rho)\eta||\vec{a}||_2, \tag{3}$$

where y are the data, $\eta \geqslant 0$ is 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 nonzero coefficients. For every point on the (η,ρ) grid, the "elastic net," one can obtain a single optimal model using a standard solver such as 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.

B. Multiobjective fitness

As mentioned in Sec. II, 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 Ockham's razor ("law of parsimony"), a simple solution is considered superior to a complicated one [38,39] 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:

$$\Gamma = \sum_{i}^{N} w_i \Gamma_i, \tag{4}$$

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 Eq. (3). This approach has three major flaws:

(1) One needs to determine suitable (problem dependent) w_i.

(2) One does not account for nonlinear trade-offs (e.g., all-or-nothing in one objective).

(3) Instead of a 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 the set of nondominated 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 in Fig. 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. Interestingly, the concept of nondomination 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

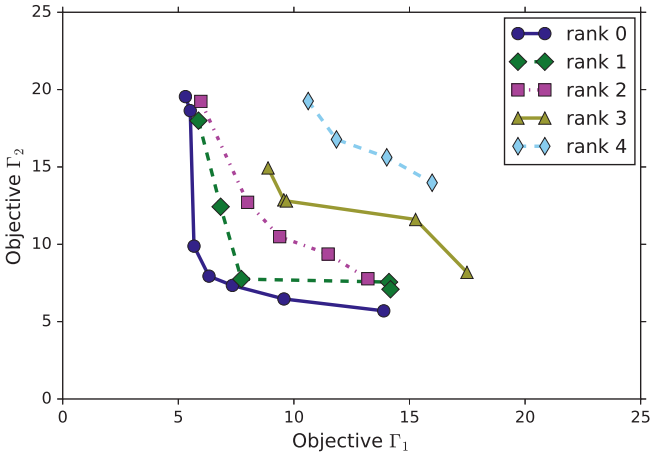


FIG. 3. Illustration of the ranking for Pareto optimization. For a given set of points in the "objectives plane" (Γ_1,Γ_2) one labels first all points with rank zero that possess at least one minimum objective, the Pareto front (filled circles, solid line). Then, these points are eliminated from the 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.

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 [40–42]. 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 nondominated sorting algorithm II (NSGAII) [40] 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

$$f_1 \succ f_2 \Longleftarrow \begin{cases} \text{rank}(f_1) < \text{rank}(f_2), \\ \text{rank}(f_1) = \text{rank}(f_2) \quad \text{and} \\ \text{distance}(f_1) > \text{distance}(f_2). \end{cases} \tag{5}$$

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 algorithm 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 [43,44].

III. OUR GP SETUP

For all applications presented in Sec. IV, 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:

$$\Gamma_1 = \Gamma_{\text{NRMSE}}(y, \hat{y}) = \frac{\sqrt{\sum_{i=1}^{N} \frac{(y_i - \hat{y}_i)^2}{N}}}{y_{\max} - y_{\min}}, \tag{6}$$

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

$$\Gamma_2 = \text{size}(f) \tag{7}$$

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 $\Gamma_1 \leqslant 0.05$ and measured data might be modeled accurately if $\Gamma_1 \leqslant 0.20$. 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 premature 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 [43,44], 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:

$$\Gamma_1 = \Gamma_{\mathrm{NRMSE}}(y, f(\vec{x},\vec{c}^*)) \tag{8}$$

with

$$\vec{c}^* = \underset{\vec{c}}{\operatorname{argmin}}\, \Gamma_{\mathrm{NRMSE}}(y, f(\vec{x},\vec{c})). \tag{9}$$

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 formulas. This is useful, since a formula (phenotype, macrostate) may be represented by many different expression trees (genotypes, microstates).

IV. 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 [45].

A. 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

$$\dot{x} = y, \tag{10}$$

$$\dot{y} = -\omega^2 x, \tag{11}$$

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.

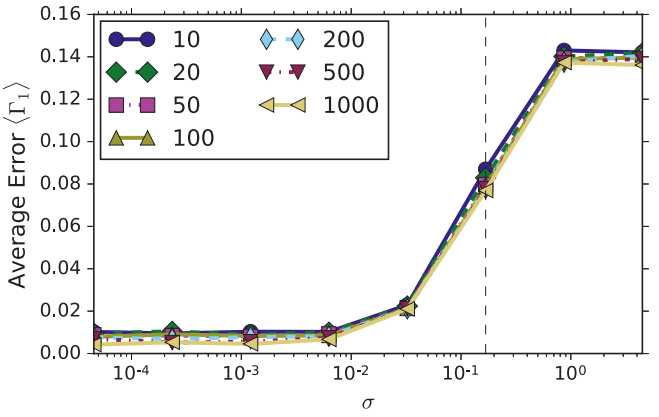


FIG. 4. Harmonic oscillator study, method FFX: NRMSE (6) versus noise level σ for different training set lengths N and fixed $\tau = 10$. Sufficiently small noise does not worsen the predictability; i.e., the prediction algorithm stops at the target training NRMSE of 1%. After 0.3 the error does not increase further, since the noise covers the signal completely. Dashed line marks $\sigma = 0.17$.

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 the subject of ongoing research. Nevertheless, a preview with qualitatively similar results can be found in Fig. 16. In Fig. 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 2. 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. 16 and Fig 6. Strong noise covers the signal and thus the error saturates.

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. 4 is shown in Fig. 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 Eq. (7) of the final model as a function of the

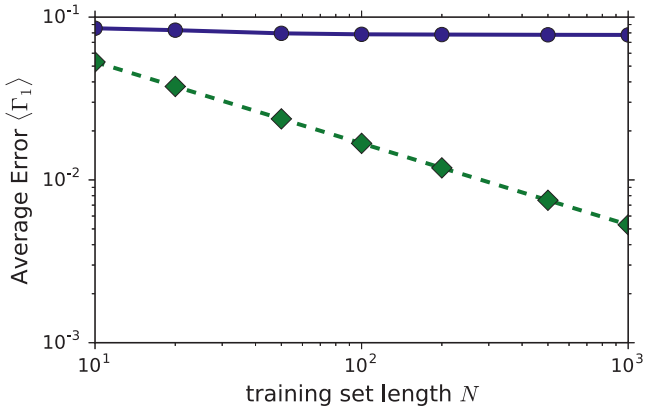


FIG. 5. Harmonic oscillator study, method FFX: In solid blue: normalized root mean squared error vs training set length N for $\sigma = 0.17$. Dashed green: $e^{-2}/\sqrt{N}$. The error decreases slightly with N, but the scaling is much less rapid than $1/\sqrt{N}$.

noise amplitude, is shown in Fig. 6. As evident we can recover the three regimes of Fig. 4. For small noise, the analytical and numerical solutions 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 a single constant; i.e., for high σ the complexity tends to smaller values as seen in Fig. 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.

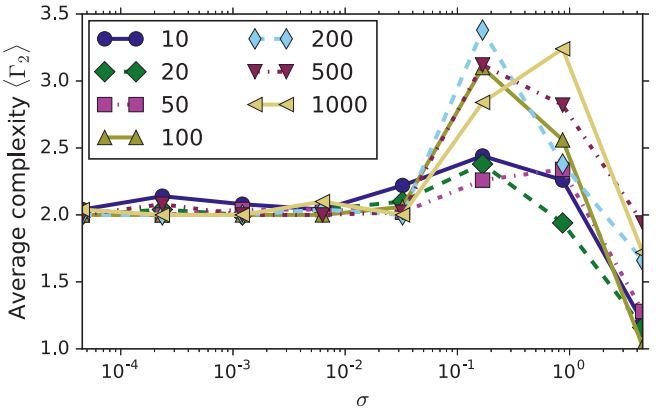


FIG. 6. Harmonic oscillator study, method FFX: Average complexity of the chosen model vs noise amplitude σ. The form-free structure allows for overfitting. For small noise, the true solution with complexity 2 is found; for higher noise levels, the algorithm starts to fit the noise and more terms are added, reflected by a higher complexity.

B. 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-Nagumo oscillators [46] 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 [47], 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 such as 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 ith oscillator is

$$\dot{v}_i = v_i - \frac{v_i^3}{3} - w_i + I_i + D\sum_{i,j} A_{ij}(v_j - v_i), \quad (12)$$

$$\dot{w}_i = \varepsilon(v_i + a - bw_i), \quad (13)$$

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 spatiotemporal 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 spatiotemporal 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 nonlocal 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 $I_i(t) = 5\chi_{t-\lfloor t \rfloor \leqslant 0.4}(t)\chi_{t \leqslant 40}(t)\chi_{i \in \{-50,-49\}}(i)$. 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

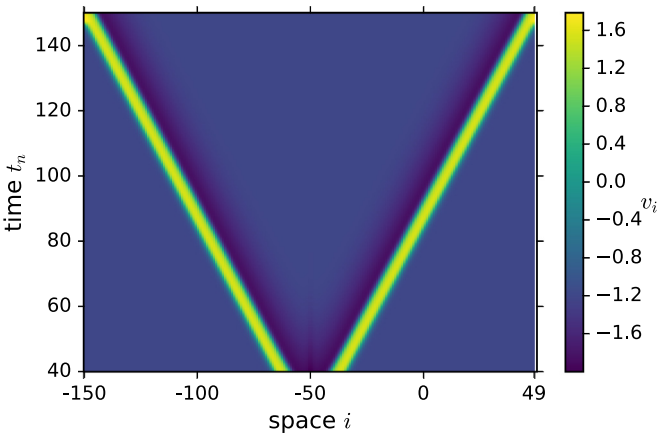


FIG. 7. Space-time plot of the pulse evolution. v_i is color coded. The front velocity is $v_f = 1.28$. Pulse width (full width half maximum) $\tau_P = 8.4$.

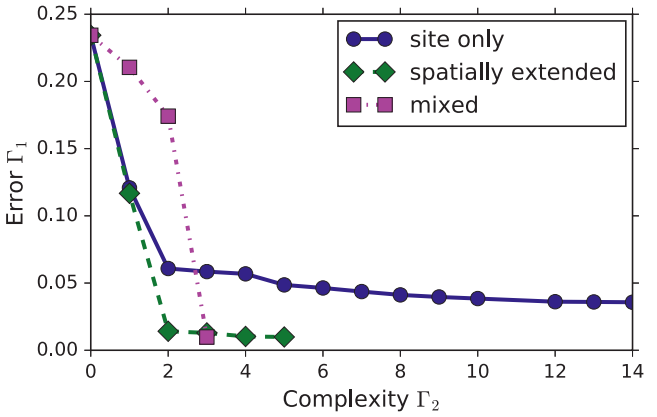


FIG. 8. Coupled spiking oscillators, method FFX: Pareto fronts for the different spatiotemporal samplings of the network data. For this plot we use $\rho = 0.95$. This leads to the nonconvex shape of the front based on the most information. The models are reevaluated on the testing set.

oscillator ($\tau_{\text{fast}} = 1$, $\tau_{\text{slow}} = \frac{1}{\varepsilon}$), while two more are due to diffusive coupling ($\tau_{\text{Diff}} = D$) 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. 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 $v_{\text{front}} \sim \xi/\tau_P = 1.28$. An animation of this can be found in the Supplemental Material [51]. The training data were recorded in three different ways:

(1) *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 $\Delta n \Delta t = -1, -2, -3, -4$.

(2) *spatially extended*. We record v_0 and additionally v_i with $i = -2, -4, \ldots, -10, -20$ (upstream direction).

(3) *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 usage 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 enough 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.

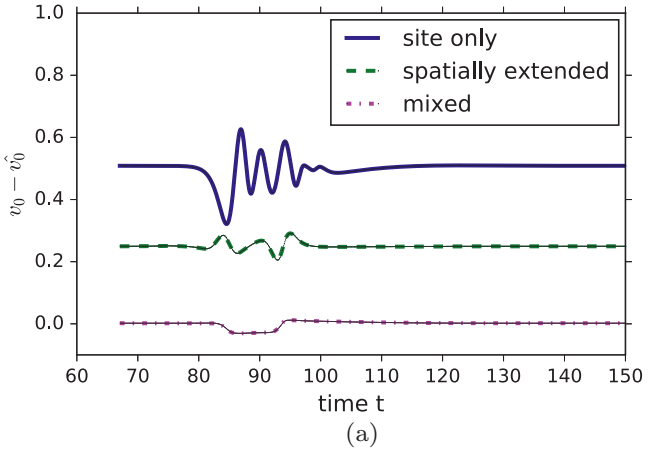
1. FFX results

We first discuss the results obtained by FFX (Sec. II A 2). In Fig. 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 data sets the algorithm has the possibility to find a combination of spatial and 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 Eq. (3). More information should not lead to a decrease in predictability. Thus, the Pareto front of a data set richer in features dominates the corresponding Pareto front of a data set with less features. Counterintuitively, using $\rho = 0.95$ [48] the front for the mixed data set becomes nonconvex 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.

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. 9(a) 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 spatiotemporal set models best the passing spike. We note that in a direct confrontation, the true and modeled signal would be hard to distinguish. In Fig. 9(b) we confront the time derivative for the model from mixed variables. The true and modeled spike are indistinguishable by eye.

The formulas of the most accurate models are shown in Table I. 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, spatiotemporal embedding is best, namely sampling the



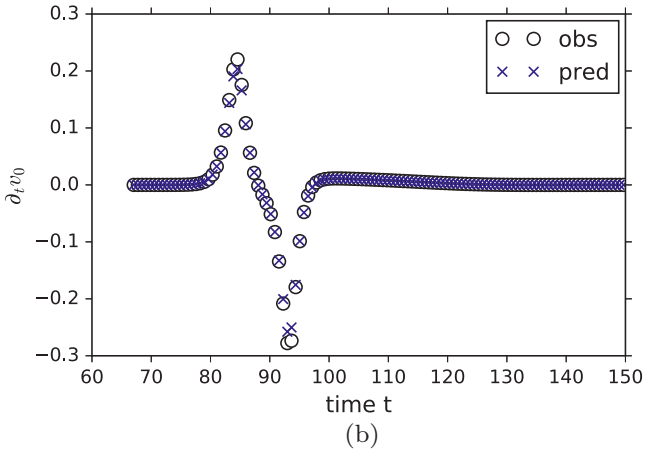


FIG. 9. Coupled spiking oscillators, method FFX. For each feature set, the most accurate model is used as the predictor $\hat{v}_0$. In (a) we show the difference $\delta v_0 = v_0 - \hat{v}_0$. The upper two curves are shifted by 0.25 and 0.5, respectively. In (b) we compare the time derivative (approximated by the finite difference quotient) of the most accurate model overall (mixed) and the real data. To better visualize the curves we only plot every seventh point. For details see text.

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 I. 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 I can be

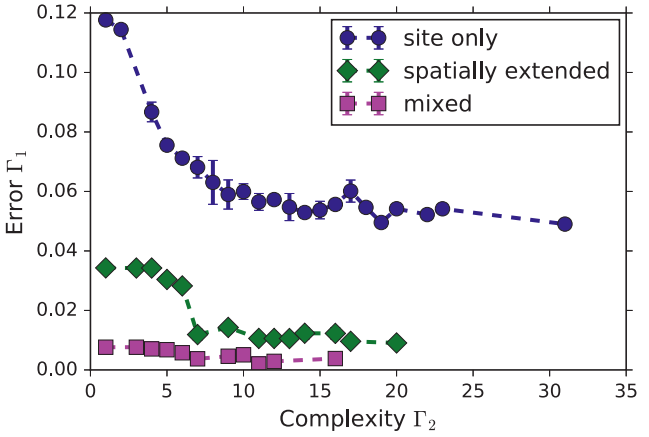


FIG. 10. Coupled spiking oscillators, method GP. Averaged Pareto fronts; for each spatiotemporal sampling option, 10 runs are conducted and the resulting complexities and errors are averaged. Error bars represent the standard deviation. For the spatially extended and mixed data sets the errors are smaller than the circle size. The models are reevaluated on the testing set.

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.

2. GP results

We again examine the Pareto-optimal models illustrated in Fig. 10. For each feature set we obtain a nonconvex 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 nondominated 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 result of Table II 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. 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.

TABLE I. Coupled spiking oscillators, method FFX. Formulas of the most accurate models. The spatiotemporal embedding reproduces the data very well; i.e., an early detection is possible.

Temporal site only	$-0.0273 + 3.34 v_{0,0} - 2.41 v_{0,0} v_{0,-10} - 2.09 v_{0,-40} v_{0,-10} + 1.64 v_{0,-20}^2 - 1.53 v_{0,-20} - 1.16 v_{0,-10} + 0.991 v_{0,-30}^2$ $+ 0.684 v_{0,0}^2 + 0.463 v_{0,-30} + 0.433 v_{0,-20} v_{0,0} + 0.373 v_{0,-20} v_{0,-10} - 0.359 v_{0,-40}^2 + 0.216 v_{0,-40} + 0.00286 v_{0,-10}^2$
Spatially extended	$-0.002\,47 + 0.897 v_{-2,0} + 0.178 v_{0,0} - 0.0650 v_{-4,0} + 0.002\,80 v_{-10,0} - 0.002\,10 v_{-8,0}$
Temporal spatial	$0.008\,94 + 0.442 v_{-4,-30} + 0.346 v_{-2,-10} + 0.175 v_{-2,0}$

TABLE II. Coupled spiking oscillators, method GP. Formulas of the most accurate models for seed 42.

Temporal site only	$v_{0,0}^2/[v_{0,-10} + \sqrt{}(-v_{0,-10}(v_{0,0} - v_{0,-30})\{v_{0,-30}/\sin(v_{0,-10} + v_{0,-20}) + \exp(v_{0,-30}) - [\sin(v_{0,-30})]^{1/2} + \cos[(v_{0,-30})^{1/2} v_{0,-40}]\})]$
Spatially extended	$0.208 v_{0,0} + 0.792 v_{-2,0} + 0.0274\,362\,547\,430\,272 \exp(-v_{-4,0}) \sin(v_{-2,0})$
Temporal spatial	$0.878 v_{-4,-30} + 0.124\,496 v_{-4,-40}$

The models provided by the GP algorithm with seed 42 are given in Table II. Due to the very general character of GP these can be overwhelming at first glance. However, we can simplify them down by using computer algebra systems such as SYMPY [24] 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 nondeterministic and overfitting result, mirrored by the high complexity of the functions involved. For spatially extended models, we obtain linear and sinusoidal components, and the 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 spatiotemporal 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.

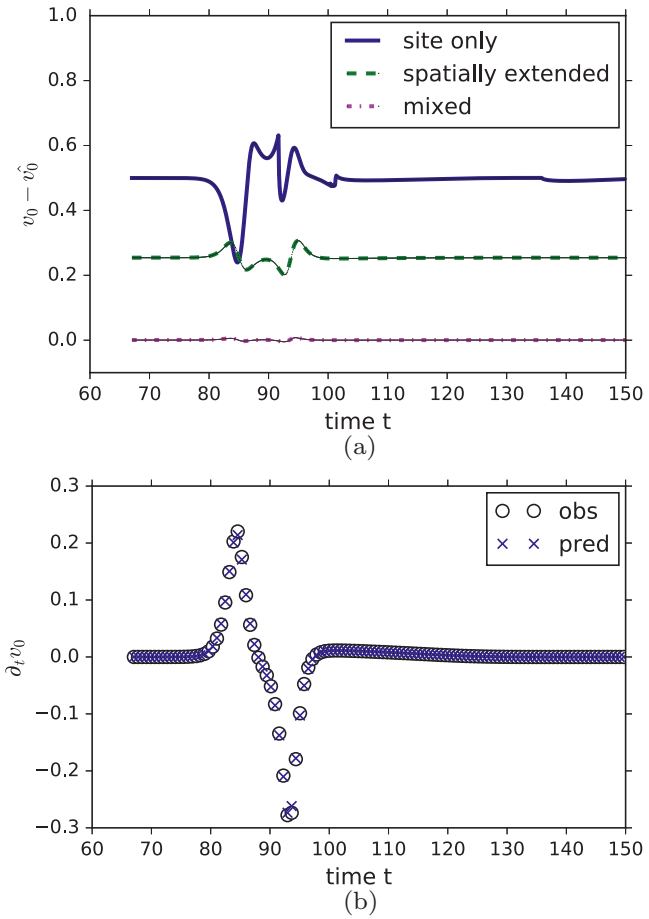


FIG. 11. Coupled spiking oscillators, method GP. For each feature set, the most accurate model is used as the predictor $\hat{v}_0$. In (a) we show the difference $\delta v_0 = v_0 - \hat{v}_0$. The upper two curves are shifted by 0.25 and 0.5, respectively. In (b) we compare the time derivative (approximated 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. To better visualize the curves we only plot every seventh point.

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.

C. 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 [45]. We join the solar power data with meteorological forecast data from the freely available European Centre for Medium-Range Weather Forecasts (ECMWF) portal [49] 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

$$\begin{aligned} \Gamma_1 &= \Gamma_{\mathrm{NRMSE}}(P(t+\tau), \hat{P}(P(t), W(t), \hat{W}(t+\tau))), \\ \Gamma_2 &= \mathrm{size}(f), \end{aligned} \tag{14}$$

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 data sets with a sampling of 1 h. While additional information from the solar power data remains unused, the prediction variables have to be interpolated. The quality of the

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

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

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 III. Furthermore, we 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 [49].

1. FFX results

The results of the FFX method are shown in Figs. 12 and 13 and the models in Table IV. 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. 13. One may wonder why a constant (complexity zero) explains about 30% of the data (Fig. 12). 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 IV, 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.

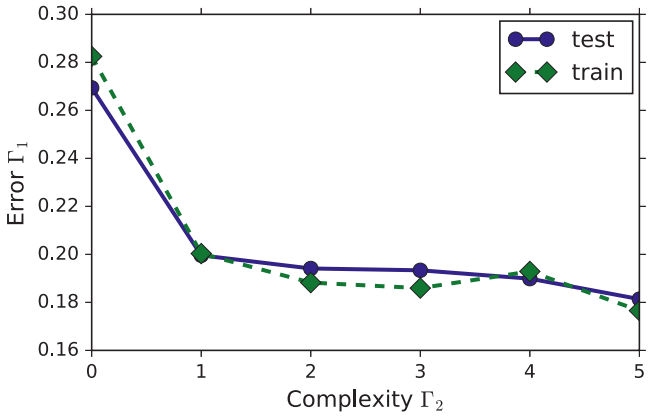


FIG. 12. Solar power study, Pareto front obtained using FFX. The results for FFX are as accurate as the ones obtained with GP. Test and training set are, however, nicely aligned. This demonstrates not only consistency of the models, but less variability of the models found.

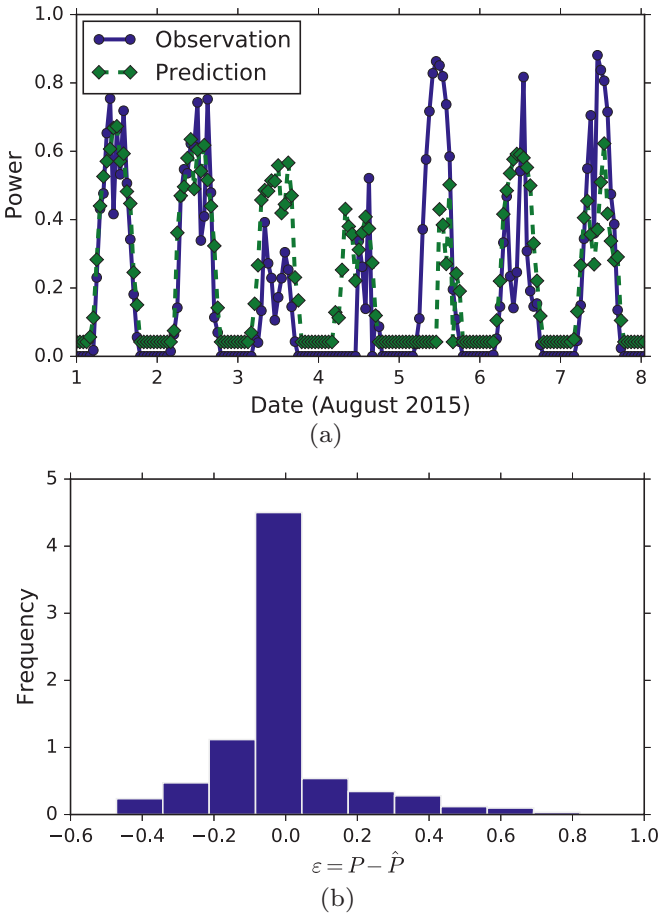


FIG. 13. Solar power study, method FFX. (a) Time series of the predicted ($\hat{P}$), and observed (P) data. We display the results of the first week of August 2015. Similarly to the GP prediction extrema are not particularly well predicted. For the linear model, even the zero values are not well hit. 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.

TABLE IV. Solar power study, method FFX: formulas of the Pareto front models.

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	$0.301 - 1.25\max(0,0.333 - x_1)\max(0,0.200 - x_1) - 0.810\max(0,0.467 - x_1)\max(0,0.200 - x_1) + 0.457x_0x_1 - 0.252\max(0,0.333 - x_1) - 0.0794\max(0,0.200 - x_1)$

2. *GP results*

Let us consider the results of our forecasting with GP shown in Fig. 14. The Pareto fronts are shown for both the training and testing set. As presented for the coupled oscillators in Sec. IV B 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 clarify this question, we show the functions found as a solution of our procedure with increasing complexity and one specific seed (42) in Table V.

From Table V 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. 15(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. 15(b), where we recognize

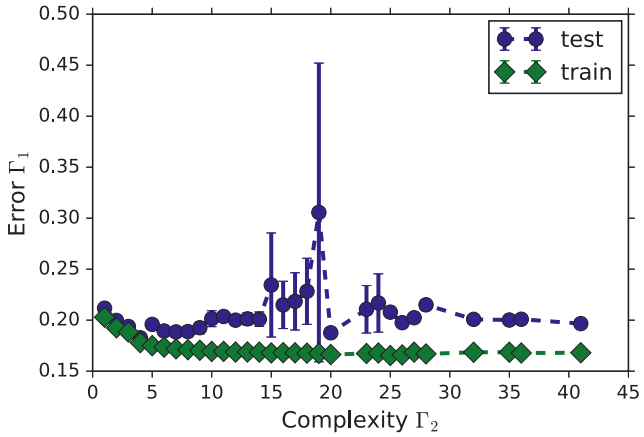


FIG. 14. Solar power study, average Pareto front obtained using GP: with increasing complexity, training and testing data first behave similarly, then testing deviates strongly indicating overfitting or too small testing data set, respectively. The peaks around complexity 20 are due to two reasons: there are only a few models on the fronts (1–3), and one of them is an extreme outlier.

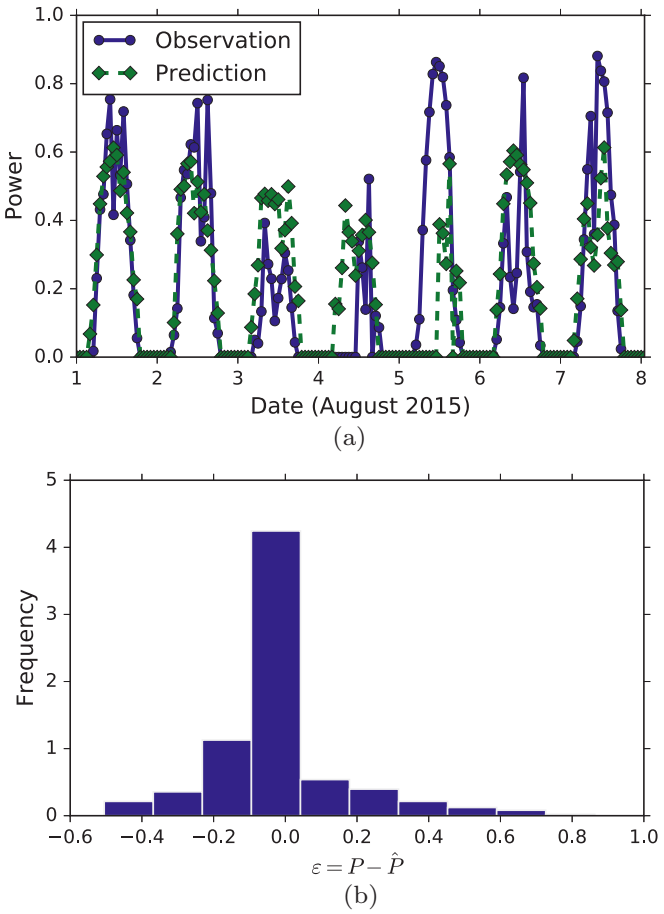


FIG. 15. Solar power study, method GP. (a) Real and predicted time series. We display the results of the first week of August 2015. Prediction used model of complexity 4 which had lowest error on the test set. (b) Histogram of the residuals $\varepsilon = P - \hat{P}$. The distribution is asymmetric around zero. The model tends to underpredict.

TABLE V. Solar power study, method GP: formulas of the Pareto front models for seed 42.

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	$x_0\{_{\sqrt{}}(x_1) + (-x_3 + 0.715)[\sin(x_1) + \sin(x_2 x_4)] \cos(x_1)\}$

an asymmetric error distribution with more probable underthan overprediction.

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 a 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.

V. 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 spatiotemporal scale and then identifying 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—and a collection of excitable oscillators where we demonstrated that the methods can perform a kind of multiscale analysis based on the data. Third, 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 [50]. 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

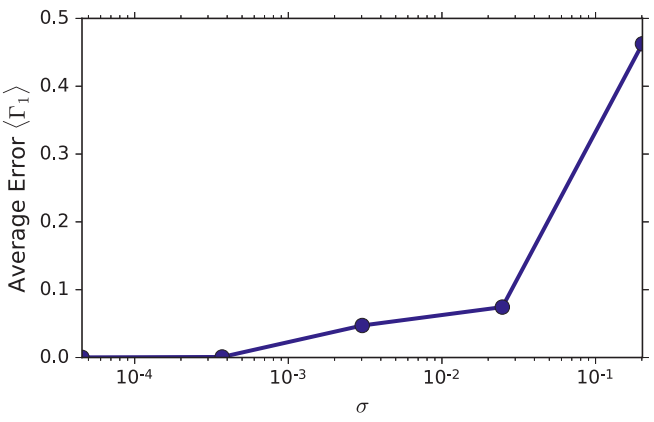


FIG. 16. Harmonic oscillator study, method GP: NRMSE (6) versus noise level σ. We fixed $N = 50$ and $\tau = 10$. The results are averaged 30 times. The absolute values for the NRMSE are higher compared to the results obtained with FFX (Fig. 16).

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.

ACKNOWLEDGMENTS

We acknowledge discussion with C. Cornaro and M. Pierro on solar power forecasts, and helpful words on machine learning by M. Segond. We thank the Unisolar Initiative for their provision of the data. M.Q. and M.A. acknowledge support by the German Federal Ministry for Economic Affairs and Energy, the ZIM project "Green Energy Forecast," Grant No. KF2768302ED4.

APPENDIX: HARMONIC OSCILLATOR NOISE STUDY WITH GP

An analogous case study to that presented in Sec. IV A using GO requires substantial code development and computational time, and is the subject of ongoing research. For completeness, we depict a small case study in Fig. 16.

[1] E. S. Sarachik and M. A. Cane, *The El Niño-Southern Oscillation Phenomenon* (Cambridge University Press, Cambridge, UK, 2010).
[2] C. F. Gauss, *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium Auctore Carolo Friderico Gauss* (sumtibus Frid. Perthes et IH Besser, Hamburg, 1809).
[3] E. Schrödinger, Phys. Rev. **28**, 1049 (1926).
[4] L. Cardelli and P. Wegner, ACM Comput. Surv. (CSUR) **17**, 471 (1985).
[5] H. U. Voss, P. Kolodner, M. Abel, and J. Kurths, Phys. Rev. Lett. **83**, 3422 (1999).
[6] M. Abel, K. Ahnert, J. Kurths, and S. Mandelj, Phys. Rev. E **71**, 015203 (2005).
[7] M. Abel, Int. J. Bifurcation Chaos **14**, 2027 (2004).
[8] D. Rey, M. Eldridge, M. Kostuk, H. D. Abarbanel, J. Schumann-Bischoff, and U. Parlitz, Phys. Lett. A **378**, 869 (2014).
[9] S. L. Brunton, J. L. Proctor, and J. N. Kutz, Proc. Natl. Acad. Sci. U.S.A. **113**, 3932 (2016).
[10] F. Takens, *Detecting Strange Attractors in Turbulence*, Lecture Notes in Mathematics, No. 898 (Springer, Berlin, 1981).
[11] T. Sauer, J. Yorke, and M. Casdagli, J. Stat. Phys. **65**, 579 (1991).
[12] K. Ahnert and M. Abel, Comput. Phys. Commun. **177**, 764 (2007).
[13] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, Science **290**, 2319 (2000).
[14] A. Singer, R. Erban, I. G. Kevrekidis, and R. R. Coifman, Proc. Natl. Acad. Sci. U.S.A. **106**, 16090 (2009).
[15] S. T. Roweis and L. K. Saul, Science **290**, 2323 (2000).
[16] M. Schmidt and H. Lipson, Science **324**, 81 (2009).
[17] J. Bongard and H. Lipson, Proc. Natl. Acad. Sci. U.S.A. **104**, 9943 (2007).
[18] M. D. Schmidt, R. R. Vallabhajosyula, J. W. Jenkins, J. E. Hood, A. S. Soni, J. P. Wikswo, and H. Lipson, Phys. Biol. **8**, 055011 (2011).
[19] K. Rodriguez-Vazquez and P. J. Fleming, Electron. Lett. **34**, 930 (1998).
[20] N. Gautier, J.-L. Aider, T. Duriez, B. Noack, M. Segond, and M. Abel, J. Fluid Mech. **770**, 442 (2015).
[21] S. L. Brunton and B. R. Noack, Appl. Mech. Rev. **67**, 050801 (2015).
[22] T. McConaghy, in *Genetic Programming Theory and Practice IX* (Springer, New York, 2011), pp. 235–260.
[23] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, J. Mach. Learn. Res. **13**, 2171 (2012).
[24] SymPy: Python Library for Symbolic Mathematics, http://www.sympy.org.
[25] E. Lorenzo, *Solar Electricity: Engineering of Photovoltaic Systems* (PROGENSA, Sevilla, 1994).
[26] E. N. Lorenz, J. Atmos. Sci. **20**, 130 (1963).
[27] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis* (Cambridge University Press, Cambridge, UK, 1997).
[28] E. Ott, T. Sauer, and J. Yorke, *Coping with Chaos*, Series in Nonlinear Science (Wiley, New York, 1994).
[29] S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering* (Westview Press, Philadelphia, 2014).
[30] H. Whitney, Ann. Math. **37**, 645 (1936).
[31] *Accuracy (Trueness and Precision) of Measurement Methods and Results. Part 1. General Principles and Definitions* (International Organization for Standardization, Geneva, 1994).
[32] H. D. I. Abarbanel, M. E. Gilpin, and M. Rotenberg, *Analysis Of Observed Chaotic Data* (Springer, New York, 1997).
[33] N. Gershenfeld, *The Nature of Mathematical Modeling* (Cambridge University Press, Cambridge, UK, 1999).
[34] C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer, New York, 2006).
[35] G. Dantzig, R. Cottle, and Y. Aneja, *Mathematical Programming: Essays in Honor of George B. Dantzig* (North-Holland, Amsterdam, 1985).
[36] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Vol. 1 (MIT Press, Cambridge, MA, 1992).
[37] H. Zou and T. Hastie, J. R. Stat. Soc., Ser. B, Stat. Methodol. **67**, 301 (2005).
[38] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, Inform. Process. Lett. **24**, 377 (1987).
[39] W. H. Jefferys and J. O. Berger, Am. Sci. **80**, 64 (1992).
[40] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, IEEE Trans. Evol. Comput. **6**, 182 (2002).
[41] E. Zitzler, M. Laumanns, and L. Thiele, *Evolutionary Methods for Design, Optimisation, and Control* (CIMNE, Barcelona, Spain, 2002), pp. 95–100.
[42] J. Knowles and D. Corne, in *Proceedings of the 1999 Congress on Evolutionary Computation* (IEEE, New York, 1999).

[43] G. F. Smits and M. Kotanchek, in *Genetic Programming Theory and Practice II* (Springer, New York, 2005), pp. 283–299.
[44] E. J. Vladislavleva, G. F. Smits, and D. Den Hertog, IEEE Trans. Evol. Comput. **13**, 333 (2009).
[45] T. Felbinger, Unisolar, Energie Sans Souci, http://www.unisolar-potsdam.de.
[46] J. Nagumo, S. Arimoto, and S. Yoshizawa, Proc. IRE **50**, 2061 (1962).
[47] A. L. Hodgkin and A. F. Huxley, J. Physiol. **117**, 500 (1952).
[48] This is the default value of the package which works well in most scenarios.
[49] *IFS Documentation CY41R1* (ECMWF, Reading, England, 2015).
[50] M.-A. Gardner, C. Gagné, and M. Parizeau, Genetic Programming and Evolvable Machines **16**, 455 (2015).
[51] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevE.94.012214 for an animation of the dynamics of our setup of coupled FitzHugh-Nagumo oscillators.