Machine Learning Conservation Laws of Dynamical Systems

Meskerem Abebaw Mebratie
Institut für Mathematik, Universität Kassel, 34109 Kassel

Rüdiger Nather
Fachbereich Elektrotechnik und Informatik, Universität Kassel, 34121 Kassel

Guido Falk von Rudorff
Institut für Chemie, Universität Kassel, 34109 Kassel and
Center for Interdisciplinary Nanostructure Science and Technology (CINSaT), Heinrich-Plett-Straße 40, 34132 Kassel

Werner M. Seiler
Institut für Mathematik, Universität Kassel, 34109 Kassel
(Dated: December 3, 2024)

Conservation laws are of great theoretical and practical interest. We describe a novel approach to machine learning conservation laws of finite-dimensional dynamical systems using trajectory data. It is the first such approach based on kernel methods instead of neural networks which leads to lower computational costs and requires a lower amount of training data. We propose the use of an "indeterminate" form of kernel ridge regression where the labels still have to be found by additional conditions. We use here a simple approach minimising the length of the coefficient vector to discover a single conservation law.

I. INTRODUCTION

Conservation laws of dynamical systems are of great theoretical and practical interest. In physics, many fundamental principles take the form of a conservation law with the conservation of energy probably the most prominent example. But also in biological and chemical models, conservation of mass and other conservation principles play a prominent role. On a more theoretical side, knowledge of a conservation law almost always provides important insight into a system. Practically, conservation laws may e.g. allow for a model reduction, if certain degrees of freedom can be expressed through others and thus be eliminated from the model equations.

Recently, there have been some efforts in discovering conservation laws of dynamical systems via machine learning, see e.g. [1–10] and references therein. In these works, quite different characterizations of conservation laws and quite different techniques from machine learning are used. Some approaches identify only a single conservation law, others try to find all of them. Most approaches are based on trajectory data, i. e. they do not require knowledge of the dynamical system (1) itself — they are "model-agnostic", as the authors of [7] call it —, but [5] uses explicitly the dynamical system and no trajectory data. Common to all these works is the use of neural networks as basic technology and that only well-known textbook examples of conservation laws have been "discovered". A conservation law is usually also found as a neural network approximating it. Most of the cited works then try in a subsequent symbolic regression step to obtain a closed-form expression for the conservation law. Judging from the presented examples, this seems to work pretty well for conservation laws which are essentially rational functions, but difficulties arise when transcendental functions with non-trivial arguments appear.

In this article, we present a novel approach to machine learning conservation laws using *kernel methods*, more precisely *kernel ridge regression* [11–13]. On the downside, this implies that we can only discover conservation laws living in the Hilbert space defined by the used kernel. Our basic tool is the inhomogeneous polynomial kernel meaning that we search mainly for polynomial conservation laws, but we will discuss how this restriction can be relaxed by extending the feature vector. On the upside, kernel methods always provide us with explicit closed-form expressions for the discovered conservation laws so that no subsequent symbolic regression is necessary. We furthermore believe that such an approach is not only computationally much more efficient than neural networks, it also requires much less training data. This point becomes important, if one is not working with synthetic numerical data (as we will do throughout this paper), but with experimental data.

The paper is organized as follows: In Section II, we briefly recall the basic properties of conservation laws of dynamical systems. Section III introduces our concept of an indeterminate regression, the key component of our method. We explain the process of discovering a single polynomial conservation law, discuss its implementation and possible validation strategies, and finally present some examples. In Section IV, we extend our analysis to the discovery of several, functionally independent conservation laws and non-polynomial conservation laws. Section V addresses some complexity considerations of our method and Section VI explores further applications of our method, including its applicability to discrete dynamical systems and implicitization problems. Finally, some conclusions are given.

II. CONSERVATION LAWS

In this work, we will be dealing with (continuous) D-dimensional dynamical systems of the form

$$\dot{x}_d = f_d(\mathbf{x}) \qquad d = 1, \ldots, D \tag{1}$$

where the vector field $\mathbf{f}$ on the right hand side is assumed to be at least once continuously differentiable (most dynamical systems in applications are at least smooth if not even analytic). We will usually assume that (1) is explicitly given, but for our basic approach this is not necessary, as it only requires knowledge of finitely many points on finitely many trajectories.

A *conservation law* or a *conserved quantity* or a *first integral* is by definition a function $\Phi\colon \mathbb{R}^D \to \mathbb{R}$ which is constant along each trajectory of the system (1). Assuming that Φ is also at least once continuously differentiable and using the Lie or orbital derivative along the vector field $\mathbf{f}$, this is equivalent to Φ satisfying the linear partial differential equation

$$f_1(\mathbf{x})\frac{\partial \Phi}{\partial x_1} + \cdots + f_D(\mathbf{x})\frac{\partial \Phi}{\partial x_D} = 0\,. \tag{2}$$

As we will be interested only in differentiable conservation laws, equation (2) provides a simple rigorous validation criterion for candidate functions Φ (if the vector field $\mathbf{f}$ is explicitly given).

Many dynamical systems in physics stem from a variational principle and their conservation laws are related to variational symmetries via Noether's theorem [14] which often allows their explicit construction with symmetry methods. By contrast, most biological models are of a more phenomenological nature without an underlying variational principle and for them it is much harder to find conservation laws. While linear ones can often be easily constructed with linear algebra (see Remark 5 below), nonlinear ones are rarely known. In principle, methods based on (adjoint) Lie symmetries allow to construct them systematically [15], but for ordinary differential equations it is usually very hard, if not impossible, to find their Lie symmetries.

A first consequence of the above definition is that any constant function Φ defines a conservation law. Obviously, such conservation laws are not useful and they are called *trivial*. In the sequel, we will only be concerned with non-trivial ones. The definition of a conservation law also implies that if the system (1) admits one, it automatically admits infinitely many. Indeed, if $\Phi_1, \ldots, \Phi_L$ are some conservation laws, then any function of the form $g(\Phi_1, \ldots, \Phi_L)$ for an arbitrary function g is a conservation law, too. If one speaks about finding "all" conservation laws, one actually means finding a maximal set of *functionally independent* conservation laws. Differentiable functions $\Phi_1, \ldots, \Phi_L$ are functionally independent, if and only if their Jacobian $\partial\mathbf{\Phi}/\partial\mathbf{x}$ has almost everywhere maximal rank meaning that the gradient vectors $\nabla_{\mathbf{x}}\Phi_1, \ldots, \nabla_{\mathbf{x}}\Phi_L$ are almost everywhere linearly independent. In physics, a Hamiltonian system of dimension $D = 2d$ admitting d functionally independent conservation laws is called *completely integrable* [16]. A dynamical system (1) may possess up to $D-1$ functionally independent conservation laws; if this is the case, each trajectory is uniquely determined by the values of these laws. A Hamiltonian system with more than d conservation laws is often called *superintegrable* [17].

We will be mainly concerned with *polynomial* conservation laws, i. e. we will search for conservation laws in the polynomial ring $\mathcal{P} = \mathbb{R}[x_1, \ldots, x_D]$. If now $\Phi_1, \ldots, \Phi_L \in \mathcal{P}$ are some conservation laws of (1), then the $\mathbb{R}$-algebra $\mathcal{A} = \mathbb{R}[\Phi_1, \ldots, \Phi_L]$ contains all polynomial conservation laws that can be constructed out of them. We note that both $\mathcal{P}$ and $\mathcal{A}$ are infinite-dimensional $\mathbb{R}$-linear spaces equipped with a natural filtration by the polynomial degree. If we truncate at a given degree q, i. e. consider only polynomials up to this degree, then we obtain finite-dimensional $\mathbb{R}$-linear spaces $\mathcal{P}_{\leq q}$ and $\mathcal{A}_{\leq q}$. This linear structure will allow us at certain places to apply methods from linear algebra.

III. INDETERMINATE REGRESSION

Our approach is based on the literal definition of a conservation law: a function Φ which is conserved, i. e. constant, along trajectories. Like most approaches presented in the literature, we therefore use numerical trajectory data and not the dynamical system (1) itself. Given finitely many points on a trajectory, any conservation law Φ must evaluate to the same value at all these points. As several trajectories may lie on the same level set of Φ, we cannot assume that at points on different trajectories Φ must evaluate to different values. We consider this situation as an *indeterminate regression problem*, i. e. a regression problem where the labels are unknown at the beginning and must be determined later by additional conditions. For the core procedure – discussed in this section – we need only the trajectory data. For certain extensions or improvements – mainly discussed in the next section – we also need the vector field $\mathbf{f}$, i. e. an explicit model of the dynamical system.

A. Discovering One Polynomial Conservation Law

We assume that we are given N points $\mathbf{x}_{m,n} \in \mathbb{R}^D$ on each of M trajectories T_m of the dynamical system (1). One could easily work with a different number of points on each trajectory. However, to avoid a bias in the regression, we prefer to take always the same number of points. Thus our indices always satisfy $m \in \{1, \ldots, M\}$ and $n \in \{1, \ldots, N\}$ and we have a total of MN data points. In addition, we introduce M yet undetermined labels y_m and our regression problem consists of finding a function Φ such that $\Phi(\mathbf{x}_{m,n}) = y_m + \epsilon_{m,n}$ with error terms $\epsilon_{m,n}$ which are minimal in a suitable sense. Note that the target value for $\Phi(\mathbf{x}_{m,n})$ does not depend on n.

This fact encodes that we search for functions which are constant along trajectories and hence that the number of labels is much smaller than the total number of data points; it also leads to a much smaller search space. Ha and Jeong [3] speak here of *grouped data*.

We use *kernel ridge regression*. Let $\mathcal{K}\colon \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ be the chosen kernel. We will throughout work with the polynomial kernel $\mathcal{K}_{c,q}(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + c)^q$ parameterised by a non-negative real number $c \geq 0$ (set in all our experiments to $c = 1$) and a degree $q \in \mathbb{N}$, although in principle any kernel would work. If K is the corresponding kernel matrix of dimension MN defined by $K_{mn,\bar{m}\bar{n}} = \mathcal{K}(\mathbf{x}_{m,n}, \mathbf{x}_{\bar{m},\bar{n}})$, then the coefficients of the unique solution of the regression problem can be expressed in closed form as

$$\boldsymbol{\alpha} = \left(K + \lambda \mathbb{1}_{MN}\right)^{-1} \mathbf{Y} \tag{3}$$

where $\mathbb{1}_{MN}$ denotes the MN-dimensional identity matrix, λ the regularization parameter of the ridge regression and $\mathbf{Y}$ the label vector. Because of our grouped data, the MN-dimensional label vector $\mathbf{Y}$ can be written as the Kronecker product of an N-dimensional vector $\mathbf{1}_N$ consisting only of ones and an M-dimensional trajectory label vector $\mathbf{y}$: $\mathbf{Y} = \mathbf{1}_N \otimes \mathbf{y}$. Here and in the sequel, double indices m, n are always sorted first by the value of m and then by the value of n. In other words, we consider first all points on the first trajectory, then all points on the second trajectory and so on. The solution itself is given by the linear combination

$$\Phi(\mathbf{x}) = \sum_{m=1}^{M} \sum_{n=1}^{N} \alpha_{m,n} \varphi_{m,n}(\mathbf{x})\,. \tag{4}$$

of the base functions $\varphi_{m,n}(\mathbf{x}) = \mathcal{K}_{c,q}(\mathbf{x}, \mathbf{x}_{m,n})$ spanning an at most MN-dimensional subspace of the Hilbert space defined by the kernel.

Because of the used of grouped data, the solution actually lies in a much smaller subspace. Consider the "compressed" matrix $\hat{K} \in \mathbb{R}^{MN \times M}$ defined by

$$\hat{K} = \left(K + \lambda \mathbb{1}_{MN}\right)^{-1} \cdot (\mathbb{1}_M \otimes \mathbb{1}_N)\,. \tag{5}$$

Its first column is the sum of the first N columns of $\left(K + \lambda \mathbb{1}_{MN}\right)^{-1}$, its second column the sum of the next N columns of this matrix and so on. It allows us to express the coefficients $\boldsymbol{\alpha}$ directly through the trajectory label vector $\mathbf{y}$, namely $\boldsymbol{\alpha} = \hat{K}\mathbf{y}$. Defining M new base functions $\boldsymbol{\kappa}(\mathbf{x}) = \hat{K}^T \boldsymbol{\varphi}(\mathbf{x})$, we can finally write the solution (4) in a form emphasising the linear dependency on the yet unknown trajectory labels $\mathbf{y}$:

$$\Phi(\mathbf{x}) = \sum_{m=1}^{M} y_m \kappa_m(\mathbf{x})\,. \tag{6}$$

It also shows that – because of the grouped data – our search space is not the whole space spanned by the base functions $\varphi_{m,n}(\mathbf{x})$, but only the at most M-dimensional subspace spanned by the functions $\kappa_m(\mathbf{x})$. Note, however, that while the base functions $\varphi_{m,n}(\mathbf{x})$ are completely determined by the data, the functions $\kappa_m(\mathbf{x})$ also depend on the regularization parameter λ.

In a classical regression, the label vector $\mathbf{Y}$ is part of the given data and the problem consists of finding a function Φ of the form (4) that fits the data as good as possible. In our indeterminate regression, the problem consists of finding trajectory labels $\mathbf{y}$ such that (6) defines a function Φ that is as constant as possible along the trajectories. We propose the following solution: choose $\mathbf{y}$ such that the coefficient vector $\boldsymbol{\alpha} = \hat{K}\mathbf{y}$ has a minimal L^2 norm. If we introduce the matrix $A = \hat{K}^T\hat{K} \in \mathbb{R}^{M \times M}$, then this requirement is equivalent to minimizing

$$F(\mathbf{y}) = \mathbf{y}^T A \mathbf{y}\,. \tag{7}$$

Since any kernel matrix K is symmetric and positive definite, A is a symmetric and positive definite matrix, too. Thus the function F possesses a unique global minimum at $\mathbf{y} = \mathbf{0}$. Since the zero function is a trivial conservation law, it is of no use of us and we must avoid this solution. To achieve this, we impose a linear constraint of the form

$$\mathbf{a}^T \mathbf{y} = 1 \tag{8}$$

with an arbitrary nonzero vector $\mathbf{a} \in \mathbb{R}^M$. This side condition can be satisfied by *any* conservation law $\Phi(\mathbf{x})$ after a shift by a constant. Indeed, assume that Φ leads to trajectory labels $\bar{\mathbf{y}}$ satisfying $\mathbf{a}^T\bar{\mathbf{y}} = c$ for some real number $c \neq 1$. Then the shifted conservation law $\tilde{\Phi}(\mathbf{x}) = \Phi(\mathbf{x}) + C$ with $C = (1-c)/\sum_{i=1}^{M} a_i$ leads to trajectory labels satisfying (8).

Thus for determining the trajectory labels $\mathbf{y}$, we must solve a quadratic program in M variables with a single linear constraint:

$$\begin{aligned} &\min_{\mathbf{y} \in \mathbb{R}^M} \ \mathbf{y}^T A \mathbf{y} \\ &\textbf{subject to } \ \mathbf{a}^T \mathbf{y} = 1\,. \end{aligned} \tag{9}$$

Such a quadratic program represents a classical problem in optimization for which many numerical solution methods are available. In fact, it is not difficult to solve (9) in closed form. A simple calculation yields its unique solution to be

$$\mathbf{y} = \frac{A^{-1}\mathbf{a}}{\mathbf{a}^T A^{-1} \mathbf{a}} \tag{10}$$

which is obviously non-trivial for any vector $\mathbf{a} \neq \mathbf{0}$. But in practise, it is more efficient and stable to use a numerical method.

This approach emerges naturally from the general philosophy of a ridge regression. Its loss function consists of two terms – the root mean squared error and the L^2 regularization – with the hyperparameter λ scaling the relative strength of the regularization. For $\lambda = 0$, one obtains the classical least squares solution which is often numerically unstable and prone to overfitting. In the

limit $\lambda \to \infty$, the solution $\boldsymbol{\alpha}$ tends towards $\mathbf{0}$ severly underfitting most functions. Comparing with Gaussian process regression, the expression of the mean prediction of a Gaussian model is identical to the prediction of a ridge regression model with identical kernel function and hyperparameters, if λ is identified as the (expected) variance of the noise of the training data labels. Therefore, λ should be vanishingly small for virtually noise-free data. In our results, this is always the case.

We avoid underfitting $\mathbf{y}$ by cross-validation and our acceptance criteria (see below), which mandate a small error on the validation sets. Large values of λ would yield more significant errors on the validation set due to overfitting. It is important to note that, while the trajectory labels $\mathbf{y}$ are chosen solely from the L^2 norm of the model coefficients $\boldsymbol{\alpha}$, the prediction accuracy on the validation set is used to choose which λ alongside its specific trajectory labels $\mathbf{y}$ is chosen.

If the kernel model is unable to model the data adequately, e. g. for a kernel function with compact support which does not match the data distribution, the model coefficients $\boldsymbol{\alpha}$ become large, requiring strong regularisation. In the limit of the kernel function being so wide that it is almost constant over the feature space of a given data set, K has almost the same value in all entries, such that all predictions approach 0, and the model coefficients can only be reduced with strong regularisation. If the kernel function is too narrow such that almost none of the data points in feature space interact (and, consequently, almost all predictions are 0), K is approximately the identity matrix, so $\boldsymbol{\alpha} \approx (1+\lambda)^{-1}\mathbf{Y} \approx \mathbf{Y}$. This case again yields a large L^2 norm of the model's coefficient vector $\boldsymbol{\alpha}$. The coefficient vector of a model that avoids both limit cases and is able to model the data with predictive power thus has a smaller L^2 norm. Since the acceptance criterion for a conservation law tests for predictive power on the validation sets, we only consider models that can learn the trajectory data and, therefore, have a coefficient vector $\boldsymbol{\alpha}$ with a low L^2 norm.

B. Practical Realization

Since we are working with grouped data, one has to choose two parameters for determining the size of used data set: the number M of trajectories and the number N of points on each trajectory. The total number of points is thus MN. As generically each trajectory provides information about a further level set of the conserved quantity, we generally prefer larger values of M and smaller values of N, but in most examples no big difference is noticable. For a reasonable sampling of the phase space, one should take $M \geq D$ with D the phase space dimension. Furthermore, the parameters M and N are chosen such that both the hold-out and the training sets in a stratified five-fold cross-validation can be filled with the same number of points (thus we typically choose values such that MN is a multiple of 25).

If the vector field $\mathbf{f}$ is explicitly known and synthetic numerical data are used, as we will do throughout this paper, then the points on each trajectory should be about equally spaced to avoid a sampling bias. This is achieved by not integrating the system (1) in the given form, but normalising $\mathbf{f}$ first. Thus we use as right hand side the field $\mathbf{f}/\|\mathbf{f}\|$. This modification does not change the trajectories, but only their time parametrization. If we now take on each trajectory points at times separated by a fixed time interval Δt, they are automatically equally spaced with respect to the arc length. The initial points for the different trajectories are randomly picked inside a rectangular cuboid and we integrate from each initial point both forward and backward.

Once the data have been produced, the regularization parameter λ and the trajectory labels $\mathbf{y}$ must be determined. Although they play very different roles – λ is the hyperparameter of the kernel ridge regression and $\mathbf{y}$ represents the solution –, we compute them simultaneously. For λ we use a grid search (first on a logarithmic grid for getting the right order of magnitude and then on a linear grid for refinement) and for $\mathbf{y}$ a 5-fold cross-validation. We first put aside as hold-out set a stratified random sample of 20% of the generated data points; stratified means here that we randomly chose on each trajectory 20% of the points on it.

The remaining 80% of the data points are randomly divided in five disjoint subsets which are again stratified, i. e. each set contains the same number of points from each trajectory. This yields five different splits where always four of the subsets are used as training set and the remaining subset as test set. For each value of λ on our grid, we determine the optimal trajectory labels $\mathbf{y}_\lambda$ by the condition that the mean value of the L^2 norms of the five coefficient vectors $\boldsymbol{\alpha}_\lambda^{(i)}$ obtained for the five different training sets is minimal. Since this choice is identical to the loss function for the individual regressions, there is no trade-off between λ and $\boldsymbol{\alpha}_\lambda^{(i)}$. For the actual computations, we first determine for each split and for the considered value of λ the matrix $A_\lambda^{(i)} = \left(\hat{K}_\lambda^{(i)}\right)^T \hat{K}_\lambda^{(i)}$ and then solve the quadratic program (9) for $A = \frac{1}{5}\sum_{i=1}^{5} A_\lambda^{(i)}$.

Each vector $\boldsymbol{\alpha}_\lambda^{(i)}$ determines a function $\Phi_\lambda^{(i)}$ which we evaluate on the test points of the ith split. The result for each point is compared with the entry of the trajectory label vector $\mathbf{y}_\lambda$ corresponding to the trajectory on which the test point lies. We then compute the root mean squared error over all test points of all splits and choose that value for λ which yields the minimal error (together with the corresponding label vector $\mathbf{y}_\lambda$).

For the chosen values of λ and $\mathbf{y}_\lambda$, we determine the final coefficient vector $\boldsymbol{\alpha}_\lambda$ using all data points outside the hold-out set which then defines our final candidate conservation law Φ_λ. The function Φ_λ is then evaluated at all points in the hold-out set and the results are again compared with the labels for the corresponding trajectories. We call the root mean squared error over all these tests ϵ_{gen} and consider it as a measure for the general-

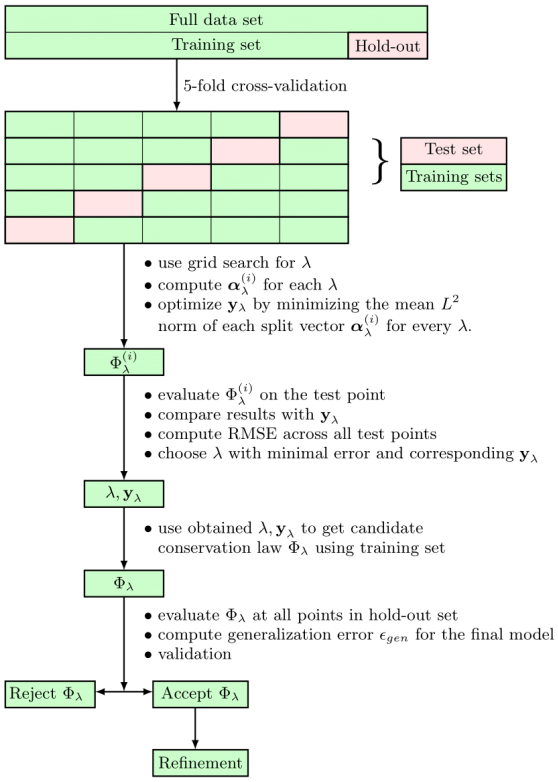


FIG. 1: Flowchart of the indeterminate regression.

ization error of our final model. Figure 1 depicts the complete procedure.

Entering the obtained trajectory labels $\mathbf{y}$ into (6) yields a candidate conservation law Φ. Since we use the polynomial kernel $\mathcal{K}_{c,q}$, this candidate is a polynomial written as a linear combination of the base functions $\kappa_m(\mathbf{x})$. As they are dense polynomials and not convenient for a human reader, we explicitly rewrite Φ as a distributed polynomial

$$\Phi(\mathbf{x}) = \sum_{0\leq|\boldsymbol{\mu}|\leq q} a_{\boldsymbol{\mu}} \mathbf{x}^{\boldsymbol{\mu}} \tag{11}$$

where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_D)$ denotes an exponent vector and the $a_{\boldsymbol{\mu}}$ are numerical coefficients determined from $\mathbf{y}$. To normalize, we divide all $a_{\boldsymbol{\mu}}$ by the one with the largest absolute value, but continue to call them $a_{\boldsymbol{\mu}}$. In all our examples, the results will be presented in this form.

Typically, most of the obtained coefficients $a_{\boldsymbol{\mu}}$ are close to zero. As the largest coefficient is 1 after the normalization, we consider all coefficients with an absolute value several orders of magnitude smaller as numerical artifacts and set them exactly zero. In our experience, a reasonable threshold is often 10^{-5} or 10^{-6}. As a "beautification", one may also think about rounding all coefficients to the corresponding number of digits, as this often helps to make coefficients exactly equal which differ only by a very small amount. But we will present here in the next section a more rigorous alternative.

C. Validation and Refinement

The above outlined procedure will always produce some candidate function Φ, but this function does not necessarily define a conservation law. In fact, the studied dynamical system (1) might not possess any conservation law at all – at least not within the Hilbert space defined by the used kernel. Thus we need a validation procedure for accepting or rejecting a candidate.

There are two fairly immediate possibilities for a purely numerical validation. One verifies directly whether Φ is constant along some trajectories which can be done independently of the dynamical system (1) or one uses the equivalent characterization of conservation laws via the partial differential equation (2) which, however, requires explicit knowledge of the vector field $\mathbf{f}$.

In the first approach, we need some grouped data not used for the indeterminate regression. One could e. g. take the data in the hold-out set or compute some points on new random trajectories, if this is possible. We denote these test data points by $\mathbf{x}'_{m,n}$ with $m = 1, \ldots, M'$ and $n = 1, \ldots, N'$. We then compute for each trajectory the mean value $\bar{\Phi}_m$ of the candidate Φ evaluated at the data points on this trajectory, i. e.

$$\bar{\Phi}_m = \frac{1}{N'} \sum_{n=1}^{N'} \Phi(\mathbf{x}'_{m,n}) \tag{12}$$

and then consider the root mean squared *relative* deviation of Φ from these mean values

$$\Delta_1(\Phi) = \sqrt{\frac{1}{M'N'} \sum_{m=1}^{M'} \sum_{n=1}^{N'} \frac{\left(\Phi(\mathbf{x}'_{m,n}) - \bar{\Phi}_m\right)^2}{\bar{\Phi}_m^2}} \tag{13}$$

We accept Φ as a conservation law, if $\Delta_1(\Phi) < \epsilon_1$ for a user determined validation threshold $\epsilon_1 > 0$.

Kernel methods provide us automatically with a candidate Φ in a symbolic form without any need for a symbolic regression. Furthermore, it is easy to compute its derivatives $\partial\Phi/\partial\mathbf{x}$. Thus, assuming that the vector field $\mathbf{f}$ is explicitly known, a direct *symbolic* validation via the partial differential equation (2) seems possible. However, as the coefficients in the representation (11) have been determined numerically, we cannot expect that (2) is satisfied exactly. Thus we resort instead again to a purely numerical approach.

This time, there is no need to use grouped data. We choose a random cloud of points $\mathbf{x}'_j \in \mathbb{R}^D$ with $j = 1, \ldots, J$ reasonably sampling some open subset of $\mathbb{R}^D$; this is almost guaranteed, if we take $J = 5D$ or $J = 10D$. Then we evaluate the left hand side of (2) at all points $\mathbf{x}'_j$ and consider the root mean squared error

$$\Delta_2(\Phi) = \sqrt{\frac{1}{J} \sum_{j=1}^{J} \left(\sum_{d=1}^{D} f_d(\mathbf{x}'_j) \frac{\partial\Phi}{\partial x_d}(\mathbf{x}'_j) \right)^2}. \tag{14}$$

We accept Φ as a conservation law, if $\Delta_2(\Phi) < \epsilon_2$ for some prescribed threshold ϵ_2.

From a mathematical point of view, these two approaches are equivalent and in experiments one indeed observes a close correlation between the two computed errors. $\Delta_1(\Phi)$ is easier to compute and as a relative variation easier to interpret which in turn makes it easier to choose the threshold ϵ_1. On the other hand, the fact that the second approach uses a random cloud of points independent of the given data is an important advantage, if data is difficult or expensive to obtain. As this was no issue in our examples, we always provide $\Delta_1(\Phi)$ computed via the hold-out set.

If the vector field $\mathbf{f}$ is explicitly known, one can use the symbolic evaluation of the partial differential equation (2) for a *refinement* of candidates having passed the numerical validation. Our starting point is the representation (11) of Φ as an expanded polynomial – after all very small coefficients have been set to zero. Let

$$\overline{\Phi}(\mathbf{x}, \boldsymbol{\delta}) = \sum_{\substack{0 \le |\boldsymbol{\mu}| \le q \\ a_{\boldsymbol{\mu}} \neq 0}} (a_{\boldsymbol{\mu}} + \delta_{\boldsymbol{\mu}}) \mathbf{x}^{\boldsymbol{\mu}} \tag{15}$$

be a perturbed form of this representation with yet undetermined perturbations $\delta_{\boldsymbol{\mu}}$. As typically many coefficients $a_{\boldsymbol{\mu}}$ in (11) will be very small, we can expect that the number of these perturbations will be much smaller than the dimension $n_{D,q} = \binom{D+q}{q}$ of the vector space of polynomials of degree at most q in D variables.

We assume now furthermore that our vector field $\mathbf{f}$ is polynomial, too, with entries of degree at most $q_{\mathbf{f}}$. Entering (15) into the left hand side of the partial differential equation (2) and expanding, we obtain a polynomial

$$\sum_{d=1}^{D} f_d(\mathbf{x}) \frac{\partial \overline{\Phi}(\mathbf{x}, \boldsymbol{\delta})}{\partial x_d} = \sum_{0 \le |\boldsymbol{\nu}| < q + q_{\mathbf{f}}} b_{\boldsymbol{\nu}}(\boldsymbol{\delta}) \mathbf{x}^{\boldsymbol{\nu}} \tag{16}$$

where the coefficients $b_{\boldsymbol{\nu}}(\boldsymbol{\delta})$ depend linearly on the perturbations $\delta_{\boldsymbol{\mu}}$, since (15) is linear in them and (2) is a linear differential equation. Any perturbation $\boldsymbol{\delta}$ with $\Delta_2\big(\overline{\Phi}(\mathbf{x}, \boldsymbol{\delta})\big) < \epsilon_2$ is consistent with the used data (obviously, here using Δ_2 is more natural than taking Δ_1).

We compute the least-squares solution $\boldsymbol{\delta}^*$ of the linear system $b_{\boldsymbol{\nu}}(\boldsymbol{\delta}) = 0$ for $0 \le |\boldsymbol{\nu}| < q + q_{\mathbf{f}}$. Note that despite its outer appearance this is an inhomogeneous linear system, as the coefficients $b_{\boldsymbol{\nu}}(\boldsymbol{\delta})$ usually contain constant terms. Generally, we can expect this system to be overdetermined, as the number of coefficients $b_{\boldsymbol{\nu}}$ is larger than the number of perturbations $\delta_{\boldsymbol{\mu}}$ for $q_{\mathbf{f}} > 1$ (i. e. for a nonlinear dynamical system). However, depending on the exact form of $\mathbf{f}$ and $\overline{\Phi}$, it may happen in exceptional cases that the system is underdetermined. Then the least-squares solution is not unique and it is natural to take the unique one of minimal norm. After possibly rounding coefficients to a prescribed number of digits, our final conservation law is then given by $\overline{\Phi}(\mathbf{x}, \boldsymbol{\delta}^*)$.

D. Examples

Most dynamical systems appearing in physics or biology depend on *parameters* and these also show up in their conservation laws. We will therefore always consider the parameters as further state space variables with trivial dynamics. Obviously, this approach increases the dimension D of the dynamical system – in biological systems often significantly. Furthermore, one generally has now to work with a kernel of higher degree q to accommodate for the dependency on the parameters.

Example 1. A classical textbook example of a physical system with a conservation law is the *Hénon-Heiles system* from astronomy:

$$\begin{aligned} \dot{q}_1 &= p_1\,, \quad \dot{q}_2 = p_2\,, \\ \dot{p}_1 &= -q_1 - 2q_1q_2\,, \quad \dot{p}_2 = -q_2 - q_1^2 + q_2^2\,. \end{aligned} \tag{17}$$

It is a Hamiltonian system and thus has as a conserved quantity the total energy given by

$$E = \frac{1}{2}(p_1^2 + p_2^2 + q_1^2 + q_2^2) + q_1^2 q_2 - \frac{1}{3} q_2^3\,. \tag{18}$$

The trajectories are rather different for different values of E. For $E < 1/6$, all trajectories are bounded and regular; for $E > 1/6$ most trajectories show a chaotic behaviour.

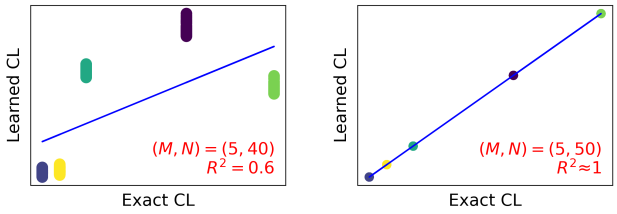


FIG. 2: Correlation coefficients for the coefficients of the Hénon-Heiles conservation law. Left: 200 data points. Right: 250 data points.

We conducted some experiments on the Hénon-Heiles system using different initial data sets, some composed entirely of points on chaotic trajectories and some using only points on regular trajectories. We were particularly interested in estimating how many data points are necessary to discover the conservation law using a cubic polynomial kernel. It turned out that with 200 data points no correct polynomial was learned, whereas with 250 data points good results were achieved (see Figure 2). This number should be contrasted with the dimension $n_{4,3} = 21$ of the space of polynomials of degree 3 in four variables: we needed roughly ten times as many data points as the number of coefficients to be determined. A further increase of the number of data points did not lead to notable improvements in the validation.

More precisely, the indeterminate regression returned for the data condition $(M, N) = (5, 50)$ and regular trajectories the following approximate conservation law (coefficients rounded to six digits) $\Phi = q_1^2 q_2 + 0.501708 q_1^2 -$

$0.336153q_2^3+0.501918q_2^2+0.501854p_1^2+0.501817p_2^2$ with a generalization error of $\epsilon_{\text{gen}} = 1.9\times10^{-7}$ and a validation error of $\Delta_1(\Phi) = 1.5\times10^{-7}$. Our refinement procedure produced here the exact expression for the energy, thus eliminating all numerical errors in the coefficients.

While the basic results were very similar for data sets consisting of regular or chaotic trajectories, respectively, one could see some small differences in a closer analysis. Under the same data condition as above, chaotic trajectories yielded $\Phi = q_1^2q_2 + 0.500001q_1^2 - 0.333334q_2^3 + 0.500001q_2^2+0.500002p_1^2+0.500001p_2^2$, which is almost the exact expression, with $\epsilon_{\text{gen}} = 3.9\times10^{-8}$ and $\Delta_1(\Phi) = 1.5\times10^{-8}$. It seems that for a fixed total number of data points it is better in the chaotic case to use a smaller number of trajectories with more points on each of them, whereas in the regular case more trajectories with a lower number of points work better. A possible explanation could be that a chaotic trajectory has a higher fractal dimension. On the one hand, it thus contains more information about the corresponding level set of the energy, but on the other hand more data points are necessary to extract this information.

Example 2. The *non-dissipative Lorenz model* is described by the three-dimensional system

$$\dot{x} = \sigma y\,, \quad \dot{y} = -xz + rx\,, \quad \dot{z} = xy \tag{19}$$

with two parameters σ, r. In contrast to the better known chaotic Lorenz model, it possesses two conservation laws:

$$\Phi_1 = \frac{1}{2}x^2 - \sigma z\,, \quad \Phi_2 = rz - \frac{1}{2}y^2 - \frac{1}{2}z^2\,. \tag{20}$$

Considering the parameters as dynamical variables, both are homogeneous quadratic polynomials and our search space has the dimension $n_{5,2} = 35$.

Using the data condition $(M, N) = (5, 10)$, we discovered the conservation law

$$\begin{aligned}\Phi = {}& 1.487557\times10^{-3}\,(-rz + 0.500004\,y^2 + 0.500004\,z^2)\\ &+ 5.455222\times10^{-3}\,(-\sigma z + 0.499999\,x^2)\end{aligned} \tag{21}$$

with a generalization error $\epsilon_{\text{gen}} = 1.411\times10^{-8}$ and a validation error $\Delta_1(\Phi) = 1.363\times10^{-8}$. Obviously, it represents in very good approximation a linear combination of Φ_1 and Φ_2. This time, the number of needed data points is only a bit larger than the number of coefficients. Figure 3a shows the conservation error of Φ along some newly computed random trajectories. Our refinement procedure transformed Φ into the exact conservation law.

There is nothing special about the found linear combination, but the data points simply select which one suits them particularly well. We repeated the experiment with new random trajectories and obtained this time approximately the linear combination $7.352196\times10^{-4}\,\Phi_1 + 5.643860\times10^{-3}\,\Phi_2$ with similar errors. Obviously, the two linear combinations are linearly independent so that in principle one could extract Φ_1 and Φ_2 out of them. However, in general this way does not represent a useful approach to discover several conservation laws. We will discuss better methods below.

IV. EXTENSIONS

A. More General Conservation Laws

If $\mathcal{H}$ is the reproducing kernel Hilbert space uniquely associated with the chosen kernel $\mathcal{K}$, then our approach searches for conservation laws only in the M-dimensional subspace of $\mathcal{H}$ of all functions of the form (6). In the case of an inhomogeneous polynomial kernel of degree d, we can thus only find conservation laws which are polynomials in $\mathbf{x}$ of maximal degree d.

If one has a priori knowledge which other, say transcendental or rational, functions may appear in a conservation law, it is rather easy to adapt our approach such that also polynomials in these functions are discovered. Assume we suspect that the functions $\eta_1(\mathbf{x}), \dots, \eta_L(\mathbf{x})$ might occur (in [6], this is called the "theorist setup" with known "basis functions"). In our basic approach, the feature vector for the kernel ridge regression coincides with the coordinate vector $\mathbf{x}$ of the dynamical system. Now, we simply extend the feature vector by the expressions $\eta_1(\mathbf{x}), \dots, \eta_L(\mathbf{x})$, before we start the kernel ridge regression. Obviously, this will increase the dimension of the feature vector from D to $D + L$ and thus the computational costs. But this approach allows us to find any conservation law that is polynomial in the variables $\mathbf{x}$ *and* the chosen transcendental functions $\boldsymbol{\eta}(\mathbf{x})$ of maximal degree d (here we have a slight difference to [6] where it is assumed that a *linear* basis of the considered function space is known).

Example 3. It is well-known that many *Lotka–Volterra systems* admit conservation laws containing *logarithmic terms* [18] (this is also very common in chemical reaction networks, as the entropy is a logarithmic quantity [19]). This is easily accommodated by extending the feature vector: we add for each coordinate x_d its logarithm $\ln x_d$, i. e. we use $\big(\mathbf{x}, \ln\mathbf{x}\big)$ as feature vector for the regression. For general dynamical systems, one might worry what happens for negative x_d. But most biological or chemical models are positive and for them only the positive orthant $\mathbf{x} \in \mathbb{R}^D_{>0}$ is relevant as state space.

As a concrete example, we consider the following four-dimensional Lotka–Volterra system

$$\begin{aligned}\dot{x}_1 = x_1(3 - x_2 - x_3 - x_4)\,, \quad \dot{x}_2 = x_2(x_1 - x_3)\,,\\ \dot{x}_3 = x_3(-1 + x_1 + x_2 - x_4)\,, \quad \dot{x}_4 = x_4(-2 + x_1 + x_3)\,,\end{aligned} \tag{22}$$

belonging to the class considered in [18, Ex. 7]. It possesses the logarithmic conservation law

$$\Phi = \sum_{d=1}^{4}(x_d - \ln x_d)\,. \tag{23}$$

As the conservation law Φ is linear in the variables $\mathbf{x}$ and $\ln \mathbf{x}$, we can work with $d = 1$ and $D = 8$ and thus have a search space of dimension $n_{8,1} = 9$. Nevertheless, we needed 100 data points ($(M, N) = (5, 20)$) to discover Φ with a generalization error $\epsilon_{\text{gen}} = 3.2 \times 10^{-7}$ and a validation error $\Delta_1(\Phi) = 1.1 \times 10^{-7}$, whereas 50 points were not sufficient. The differences between the numerical coefficients of the discovered conservation law and the exact coefficients were of the order $O(10^{-6})$; thus one may conclude that we discovered the exact conservation law. The relative variation error is shown in Figure 3b.

Example 4. The *discrete sine-Gordon equation* in the form

$$\ddot{x}_\ell = k(x_{\ell+1} - 2x_\ell + x_{\ell-1}) - g \sin x_\ell \tag{24}$$

describes a chain of pendula coupled by torsion springs [20, pp. 42–44], which among other applications is sometimes used as a simple model to describe the mechanics of DNA [21]. For simplicity, we restrict to periodic chains and set $x_{\ell+L} = x_\ell$ for some given chain length $L \in \mathbb{N}$. Rewriting (24) as a first-order system yields a system of dimension $2L$ which is Hamiltonian and thus conserves the energy

$$H = \frac{1}{2} \sum_{\ell=1}^{L} \dot{x}_\ell^2 + \sum_{\ell=1}^{L} \Big[\frac{k}{2} (x_{\ell+1} - x_\ell)^2 + g(1 - \cos x_\ell) \Big] . \tag{25}$$

As the differential equation (24) contains a trigonometric term, it is natural to expect that also its conservation laws may depend on such terms. Hence, we extended the feature vector by both $\sin x_\ell$ and $\cos x_\ell$ and thus obtained a vector of dimension $4L$. Since we also treated the two parameters k, g as dynamical variables, we have here $D = 4L+2$ and $q = 3$. We worked with a very small chain with $L = 3$, so that $n_{14,3} = 680$. With 480 data points we could not discover H, but with 600 ($(M, N) = (8, 75)$) which roughly equals the number of coefficients. With a generalization error of $\epsilon_{\text{gen}} = 1.6 \times 10^{-5}$ and a validation error of $\Delta_1(\Phi) = 9.6 \times 10^{-6}$, we obtained the following numerical conservation law:

$$\begin{aligned} \Phi = {} & -1.000029\, g \cos x_1 - 1.000010\, g \cos x_2 - g \cos x_3 \\ & + 0.999992\, k x_1^2 - k x_1 x_2 - 0.999997\, k x_1 x_3 \\ & + k x_2^2 - 1.000005\, k x_2 x_3 + 1.000002\, k x_3^2 \\ & + 0.500002\, \dot{x}_1^2 + 0.500005\, \dot{x}_2^2 + 0.500002\, \dot{x}_3^2 . \end{aligned} \tag{26}$$

As one can see, the differences to the coefficients of the exact conservation law are of $O(10^{-5})$ and are easily removed by our refinement procedure. Figure 3c shows the relative conservation error along random trajectories.

B. Discovering More Conservation Laws

A dynamical system might possess several (functionally independent) conservation laws. A simple way to discover these consists of an iteration of the approach presented in Section III A following an idea proposed by Wetzel et al. [2]. Assume that the validation has confirmed the discovery of a first conservation law Φ_1. Then we generate new trajectory data such that all trajectories lie on a level set of Φ_1, i. e. we choose for the trajectories initial data $\mathbf{x}_0^{(i)}$ for $i = 1, \ldots, M$ such that $\Phi_1(\mathbf{x}_0^{(1)}) = \cdots = \Phi_1(\mathbf{x}_0^{(M)})$. Applying our indeterminate regression to this new data, we will obtain a new candidate $\Phi_2(\mathbf{x})$. As our ansatz ensures that Φ_2 cannot be constant on all data points, Φ_2 must be functionally independent of Φ_1.

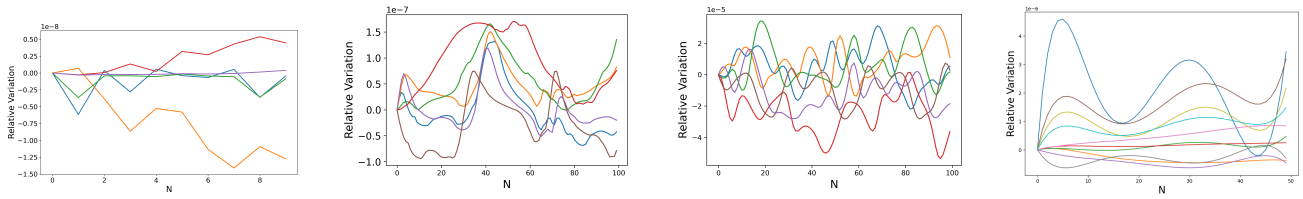
If validation confirms Φ_2 as a conservation law, we produce again new trajectory data such that all trajectories lie on common level set of Φ_1 and Φ_2. By the same reasoning as above, the new candidate Φ_3 must be functionally independent of the already found conservation laws Φ_1 and Φ_2. We continue in this manner, until the validation rejects the last candidate.

Note that this approach does not need any a priori assumption on the actual number of functionally independent conservation laws. It automatically stops, when a complete set has been found *within the considered Hilbert space*. Furthermore, this approach allows for an easy integration of a priori known conservation laws: one simply uses initial data such that all known conservation laws are constant on all initial points.

Remark 5. *Linear* conservation laws are easy to compute directly for almost any dynamical systems so that there is no need to resort to machine learning for them. Assume that (1) can be brought into the "pseudolinear" form $\dot{\mathbf{x}} = N\mathbf{g}(\mathbf{x})$ with a constant matrix $N \in \mathbb{R}^{D \times B}$ and a B-dimensional vector $\mathbf{g}$ of "building blocks". Such a structure appears e. g. naturally in chemical reaction networks where N is the stoichiometric matrix and $\mathbf{g}$ the vector of reaction rates [19, 22]. More generally, any polynomial vector field is of this form with the vector $\mathbf{g}$ consisting of all terms appearing in the vector field. One can show with elementary linear algebra that for any vector $\mathbf{v} \in \ker N^T$ the linear function $\Phi(\mathbf{x}) = \mathbf{v} \cdot \mathbf{x}$ defines a conservation law and that all linear conservation laws are of this form – see e. g. [23]. Thus, one can precompute a complete set of independent linear conservation laws and then use machine learning only for discovering additional nonlinear conservation laws functionally independent of the linear ones.

We tested this approach on a number of examples and it worked well. It is, however, only applicable, if one can easily produce additional data points for arbitrary initial conditions, and thus cannot be used with experimental data. We will therefore describe now an alternative approach for discovering multiple conservation laws using throughout the same given data set.

Assume that the given dynamical system possesses $L > 0$ functionally independent, polynomial conservation laws. Out of these, we can build infinitely many further polynomial conservation laws, by entering them



(a) Non-dissipative Lorenz system (b) 4D Lotka-Volterra system (c) Discrete sine-Gordon equation (d) FPUT lattice with $L = 5$

FIG. 3: Relative variation of the discovered conservation law along random trajectories for different examples.

into non-vanishing polynomials. Given a degree bound $q > 0$, all the polynomial conservation laws of degree at most q form a finite-dimensional linear space. A linear constraint of the form (8) defines within this linear space an affine subspace of codimension one.

The given trajectory data induces the matrix A used to define the function F via (7) and this function will have a unique minimum on the affine subspace. One may say that the corresponding conservation law is preferred by the given data; another set of data might have another preferred conservation law. Similarly, using a different linear constraint will define another affine subspace and might also lead to another preferred conservation law. But it is difficult to assert anything about the relationship between two so constructed conservation laws.

In the sequel, we assume that we are dealing exclusively with analytic functions to avoid subtilities with the concept of functional independence for smooth functions. Since we work mainly with polynomials, this assumption is trivially satisfied. If some functions $\Psi_1(\mathbf{x}), \dots, \Psi_L(\mathbf{x})$ are functionally dependent, then their gradients $\nabla\Psi_\ell(\mathbf{x})$ are linearly dependent (over a suitable ring of functions of $\mathbf{x}$) implying that at almost any point $\bar{\mathbf{x}} \in \mathbb{R}^D$ the gradient vectors $\nabla\Psi_\ell(\bar{\mathbf{x}}) \in \mathbb{R}^D$ are linearly dependent over the reals. This simple observation implies conversely that if the gradient vectors $\nabla\Psi_\ell(\bar{\mathbf{x}})$ are linearly independent at a *single* point $\bar{\mathbf{x}} \in \mathbb{R}^D$, then the analytic functions $\Psi_1(\mathbf{x}), \dots, \Psi_L(\mathbf{x})$ must be functionally independent in their complete common domain of definition. Indeed, since rank is a lower semicontinuous function, the gradients $\nabla\Psi_\ell(\mathbf{x})$ are then linearly independent in a whole neighborhood of $\bar{x}$ and in the case of analytic functions this entails that they are linearly independent in their complete common domain of definition.

The gradient vector of a function represented in the form (6) is obviously given by

$$\nabla\Phi(\mathbf{x}) = \sum_{m=1}^{M} y_m \nabla\kappa_m(\mathbf{x})\,. \tag{27}$$

Since the functions $\kappa_m(\mathbf{x})$ are explicitly known polynomials with numerical coefficients, it is easy to determine their gradients without costly techniques like automatic differentiation.

Assume that we already know a set of functionally independent conservation laws $\Psi_1(\mathbf{x}), \dots, \Psi_L(\mathbf{x})$ – either as a priori knowledge or from previous computations – and denote their Jacobian matrix by $J_{\boldsymbol{\Psi}}(\mathbf{x})$. We now choose random points $\bar{\mathbf{x}} \in \mathbb{R}^D$, until we have found one such that $\bar{J}_{\boldsymbol{\Psi}} = J_{\boldsymbol{\Psi}}(\bar{\mathbf{x}}) \in \mathbb{R}^{D\times L}$ has full rank, i. e. rank L. Because of the assumed functional independence, almost any point $\bar{\mathbf{x}}$ will work.

Analogously, we introduce the Jacobian $J_{\boldsymbol{\kappa}}(\mathbf{x})$ of the functions $\kappa_m(\mathbf{x})$ and the matrix $\bar{J}_{\boldsymbol{\kappa}} = J_{\boldsymbol{\kappa}}(\bar{\mathbf{x}}) \in \mathbb{R}^{D\times M}$. Recall that we always assume that $M \geq D$. We assume that this matrix also has full rank D, i. e. that we can find D functions $\kappa_m(\mathbf{x})$ which are functionally independent. If this was not the case, then our search space would be so small that it would not make much sense to search for further conservation laws. In other words, this would indicate that not enough data is available.

Besides (8), we impose the L orthogonality conditions

$$\nabla\Psi_\ell(\bar{\mathbf{x}}) \cdot \nabla\Phi(\bar{\mathbf{x}}) = 0\,, \qquad \ell = 1, \dots, L\,. \tag{28}$$

As discussed above, they entail that Φ is functionally independent of the given conservation laws Ψ_ℓ. Entering (27) yields the linear constraints

$$P\mathbf{y} = 0 \tag{29}$$

with the matrix $P = \bar{J}_{\boldsymbol{\Psi}}^T \bar{J}_{\boldsymbol{\kappa}} \in \mathbb{R}^{L\times M}$. It follows from the made rank assumptions that P has full rank L. Thus we arrive at the linearly constrained quadratic program

$$\begin{aligned} &\min_{\mathbf{y}\in\mathbb{R}^M} \quad \mathbf{y}^T A \mathbf{y} \\ &\textbf{subject to} \quad P\mathbf{y} = 0 \wedge \mathbf{a}^T\mathbf{y} = 1\,. \end{aligned} \tag{30}$$

Compared with (9), we have only added L further linear constraints $P\mathbf{y} = 0$.

If the unique solution of (30) passes the validation, i. e. if the function $\Phi(\mathbf{x})$ defined by (6) indeed defines a new, functionally independent conservation law, then we can rename it as $\Psi_{L+1}(\mathbf{x})$ and iterate. The iteration only requires to add one column to the matrix $\bar{J}_{\boldsymbol{\Psi}}$ leading to one additional row in the matrix P; all other quantities remain unchanged. The iteration stops, when a candidate conservation law fails the validation.

If there are problems with numerical stability, one may take a closer look at the matrices $\bar{J}_{\boldsymbol{\Psi}}$ and/or P by computing their singular value decomposition. If the Lth

singular value is rather small, then the rows of these matrices are close to being linearly dependent. In this case, it might be useful to choose a new random point $\bar{\mathbf{x}}$ which hopefully leads to a larger value.

We recall that the functions $\boldsymbol{\kappa}(\mathbf{x})$ depend not only on the data points, but also on the regularization parameter λ. Since their Jacobian is a factor of P, we can set up the quadratic program (30) only when we already know a good value for λ. In the practical application, one must therefore distinguish two different cases. If no conservation laws are known a priori, then a first conservation law is determined with our approach for a single conservation law. As part of this determination, the hyperparameter λ is tuned. For discovering further conservation laws, we now use the quadratic program (30) working throughout with the obtained value of λ. Since λ reflects the amount of noise present in our data and we continue to work with the same data, there is indeed no need to change λ.

If some conservation laws are known a priori, then we must still first perform a cross-validation for tuning λ in the same manner as in the search for single conservation law. We discard the trajectory labels $\mathbf{y}$ obtained as a by-product of this computation and now use the quadratic program (30) for searching for further conservation laws.

Our computations take place in an $\mathbb{R}$-linear space of polynomials in D variables and degree up to q. This space is of dimension $n_{D,q} = \binom{D+q}{q}$. The base functions $\varphi_{m,n}(\mathbf{x})$ or $\kappa_m(\mathbf{x})$ are dense polynomials in this space and hence it is not advisable to work with them symbolically. We now formulate certain operations via matrices.

We first represent the vector $\boldsymbol{\varphi}(\mathbf{x})$ of polynomials as a matrix $S \in \mathbb{R}^{MN \times n_{D_q}}$ where each row contains the coefficient of one polynomial $\varphi_{m,n}(\mathbf{x})$. It follows from the definition of the base functions and from the form of the polynomial kernel that, according to the multinomial formula, the entries of S are given by

$$s_{mn,\boldsymbol{\mu}} = \binom{q}{\hat{\boldsymbol{\mu}}} c^{\mu_0} \mathbf{x}_{m,n}^{\boldsymbol{\mu}} \tag{31}$$

where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_D)$ is an exponent vector of length $0 \le |\boldsymbol{\mu}| \le q$, $\mu_0 = q - |\boldsymbol{\mu}|$ and $\hat{\boldsymbol{\mu}} = (\mu_0, \mu_1, \ldots, \mu_D)$. It follows then from its definition that we can represent the vector $\boldsymbol{\kappa}(\mathbf{x})$ by the matrix $\hat{S} = \hat{K}^T S \in \mathbb{R}^{M \times n_{D,q}}$.

For our approach to discovering several conservation laws, we also need the gradients of the polynomials $\kappa_m(\mathbf{x})$. We represent the Jacobian $J_{\boldsymbol{\kappa}}(\mathbf{x})$ by a rank 3 tensor $T \in \mathbb{R}^{D \times M \times n_{D,q-1}}$ (the last dimension is only $n_{D,q-1}$, since the derivatives are polynomials of degree at most $q-1$). For an exponent vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_D)$ of length strictly less than q, we obtain as entries

$$T_{d,m,\boldsymbol{\mu}} = (\mu_d + 1) \hat{S}_{m,\boldsymbol{\mu}+1_d} \tag{32}$$

where $\boldsymbol{\mu} + 1_d = (\mu_1, \ldots, \mu_d + 1, \ldots, \mu_D)$. Given a point $\bar{\mathbf{x}} \in \mathbb{R}^D$, the entries of the Jacobian $J_{\boldsymbol{\kappa}}(\bar{\mathbf{x}})$ arise then as a simple contraction:

$$T_{d,m} = \sum_{0 \le |\boldsymbol{\mu}| < q} T_{d,m,\boldsymbol{\mu}} \bar{\mathbf{x}}^{\boldsymbol{\mu}} \,. \tag{33}$$

As we can probably assume that the conservation laws $\Psi_1(\mathbf{x}), \ldots, \Psi_L(\mathbf{x})$ are sparse polynomials, their Jacobian $J_{\boldsymbol{\Psi}}(\mathbf{x})$ could be computed symbolically and then evaluated at the point $\bar{\mathbf{x}} \in \mathbb{R}^D$. Alternatively, one could treat them in the same way as the vector $\boldsymbol{\kappa}(\mathbf{x})$.

Example 6. We continue to study Example 2, the non-dissipative Lorenz model. Above, we discovered in a first run the conservation law (21) which represents a linear combination of the two conservation laws given in (20). Applying our approach, we discover in a second run with an additional linear constraint enforcing the orthogonality to (21) the conservation law

$$\begin{aligned} \tilde{\Phi} = {} & 7.35 \times 10^{-4} (-\sigma z + 0.499998 x^2) \\ & + 5.64 \times 10^{-3} (rz - 0.4999998 y^2 - 0.5 z^2) \end{aligned} \tag{34}$$

with a generalization error $\epsilon_{\text{gen}} = 1.244 \times 10^{-8}$ and a validation error $\Delta_1(\Phi) = 1.844 \times 10^{-7}$. Obviously, it is functionally independent of (21) so that we have now obtained a complete set of functionally independent conservation laws.

Example 7. As a "scalable" example, we consider (generalized) *Fermi–Pasta–Ulam–Tsingou (FPUT) lattices* [24] described by second-order systems of the form

$$\ddot{x}_\ell = f(x_{\ell+1} - x_\ell) - f(x_\ell - x_{\ell-1}) \,. \tag{35}$$

For simplicity, we restrict again to periodic lattices, i. e. we assume that $x_{\ell+L} = x_\ell$ for some given $L \in \mathbb{N}$. It then suffices to consider only $\ell = 1, \ldots, L$ and rewriting (35) as a first-order system yields a system of dimension $D = 2L$. The obtained system is Hamiltonian so that energy is a conserved quantity:

$$H = \frac{1}{2} \sum_{\ell=1}^{L} \dot{x}_\ell^2 + \sum_{\ell=1}^{L} F(x_{\ell+1} - x_\ell) \tag{36}$$

(here F is an antiderivative of f). Furthermore, it follows immediately from the structure of (35) that for periodic lattices momentum is conserved, i. e. we have an additional linear conservation law

$$P = \sum_{\ell=1}^{L} \dot{x}_\ell \,. \tag{37}$$

For general f (or F, respectively), H and P are the only conservation laws for (35), but for special choices further ones may exist. Famous is e. g. the *Toda lattice* defined by $F(x) = ge^x$ [25, 26], as it is a completely integrable system. One usually assumes that F can be represented by a power series and the original FPUT α-model is obtained for the simplest non-trivial choice $F(x) = \frac{1}{2}kx^2 + \frac{1}{3}\alpha x^3$. In the β-model, one adds one further term $\frac{1}{4}\beta x^4$.

We used the α-model with $L = 5$ treating the parameters k and α as further state variables with trivial dynamics. Hence, the system dimension was $D = 12$ and we had to use a polynomial kernel of degree 4. A set of 200 data

points was not sufficient for discovering the conservation law, whereas 500 data points, $(M, N) = (10, 50)$, yielded good results with a generalization error $\epsilon_{\text{gen}} = 9.8 \times 10^{-7}$ and a validation $\Delta_1(\Phi) = 1.965 \times 10^{-6}$. Figure 3d shows the relative variation along some random trajectories. If this number of points is contrasted with the dimension $n_{12,4} = 1820$ of the space of polynomials in 12 variables and maximal degree 4, then one sees that the number of data points is here less than one third of the number of coefficients.

We do not exhibit the found conservation law, as it is a rather lengthy polynomial containing 135 monomials (after discarding very small coefficients). Compared to the 25 terms in H, this appears large, but is simply due to the fact that we discovered a linear combination of H, P, powers of P and products of the latter ones with the constant parameters α and k. We will discuss in the next section how one can retrieve H from this lengthy expression. For the moment, we only note that the relative errors of the coefficients that corresponds to terms appearing in H lie between $O(10^{-5})$ and $O(10^{-3})$. Thus even without a refinement we can conclude that we have discovered H with good precision.

C. Getting Sparse Conservation Laws

Assume that, using the method described above, we have finally obtained a complete, functionally independent set of conservation laws $\Psi_1, \ldots, \Psi_L$ within the Hilbert space defined by our kernel. While the functions Ψ_ℓ will typically be sparse polynomials (compared to the full dimension of the search space), the question naturally arises whether by taking linear combinations one may obtain an even sparser set.

If we assume for a moment that all the known conservation laws have the same degree, then the following simple approach will work. Let τ_n with $n = 1, \ldots, n_{D,q}$ be some enumeration of the terms $\mathbf{x}^\mu$ in D variables of degree at most q (here D is understood as the dimension of our feature vector which might be larger than the dimension of the phase space of our dynamical system). Then we write our conservation laws in the form $\Psi_\ell = \sum_{n=1}^{n_{D,q}} c_{n\ell}\tau_n$. This representation defines a matrix $C = (c_{n\ell}) \in \mathbb{R}^{n_{D,q} \times L}$. Our problem can be formalized as searching for a non-singular matrix $T \in \mathbb{R}^{L \times L}$ such that the columns of CT contain as many zeros as possible. These columns can then be interpreted as the coefficients of a new, complete, functionally independent set of conservation laws $\tilde{\Psi}_1, \ldots, \tilde{\Psi}_L$ consisting now of sparser functions.

Finding a sparsest vector in a subspace is known to be an NP-hard problem [27]. Since we do not need the optimal solution, we use a standard relaxation trick by replacing the number of non-zero terms – sometimes called ℓ_0-norm, although it is not a norm – by the ℓ_1-norm as the closest convex norm (in the theory of compressed sensing, it has been proven that under certain conditions the ℓ_1-solution is also the sparsest one [28, 29]). Now, the problem of finding one sparse vector can be transformed into a linear program.

We first search for *one* sparse affine combination of the given conservation laws via the ℓ_1 optimization problem

$$\min_{\mathbf{a} \in \mathbb{R}^L} \|C\mathbf{a}\|_1 \quad \textbf{subject to} \quad \sum_{\ell=1}^{L} a_\ell = 1\,. \tag{38}$$

The use of an affine combination avoids as usual the trivial zero solution. It is well-known that (38) can be reformulated as an equivalent linear program:

$$\begin{aligned} \min_{\substack{\mathbf{a} \in \mathbb{R}^L \\ \mathbf{w} \in \mathbb{R}^{n_{D,q}}}} \quad & \sum_{n=1}^{n_{D,q}} w_n \\ \textbf{subject to} \quad & \sum_{\ell=1}^{L} a_\ell = 1\,, \\ & -w_n \le \sum_{\ell=1}^{L} c_{n\ell} a_\ell \le w_n\,. \end{aligned} \tag{39}$$

Now assume that we have already constructed $\tilde{L} < L$ sparse conservation laws $\tilde{\Psi}_\ell = \sum_{n=1}^{n_{D,q}} \tilde{c}_{n\ell}\tau_n$ for $1 \le \ell \le \tilde{L}$. They define a matrix $\tilde{C} = (\tilde{c}_{n\ell}) \in \mathbb{R}^{n_{D,q} \times \tilde{L}}$. We augment the linear program (39) by the additional constraint $\tilde{C}^T C\mathbf{a} = 0$ ensuring that the vector $C\mathbf{a}$ representing the new conservation law is orthogonal to and thus in particular linearely independent of the already found ones. In this manner we can iteratively construct an alternative sparse set of conservation laws. Note that the use of such orthogonality constraints leads generally to suboptimal solutions, as one cannot expect that the sparsest basis is orthogonal.

Example 8. We continue Example 6, i. e. the analysis of the non-dissipative Lorenz model. We have discovered two functionally independent conservation laws of degree two each of which contains five terms whereas the conservation laws given in (20) contain only two and three terms, respectively. Applying our sparsification procedure to the discovered conservation laws yields exactly the ones presented in (20). In this particular case, we are lucky that these obviously sparsest solutions are indeed orthogonal to each other.

The situation is more difficult when polynomial conservation laws of different degrees occur. The reason is that our approach is based on linear algebra in an L-dimensional linear subspace of the polynomial ring $\mathcal{P}$ spanned by $\Psi_1, \ldots, \Psi_L$ whereas in fact the sparsification problem lives in the truncated $\mathbb{R}$-algebra $\mathcal{A} = \mathbb{R}[\Psi_1, \ldots, \Psi_L]_{\le q}$ with q the maximal degree of the known conservation laws. The dimension of $\mathcal{A}$ is typically considerably higher than L and therefore it does usually not suffice simply to compute L sparse vectors in $\mathcal{A}$, as these will typically *not* generate $\mathcal{A}$ as an algebra.

We propose the following modification of the basic approach outlined above. Our goal is to find new conservation laws $\tilde{\Psi}_1, \dots, \tilde{\Psi}_L$ which are sparser but still generate the same algebra: $\mathbb{R}[\Psi_1, \dots, \Psi_L] = \mathbb{R}[\tilde{\Psi}_1, \dots, \tilde{\Psi}_L]$. We assume that $\Psi_1, \dots, \Psi_L$ are ordered ascendingly by degree and denote by q_ℓ the degree of Ψ_ℓ. Generally, not all degrees q_ℓ are different. Let the indizes $L_1, \dots, L_R$ define the "jump positions":

$$q_1 = \dots = q_{L_1} < q_{L_1+1} = \dots = q_{L_2} < \dots \\ \dots < q_{L_{R-1}+1} = \dots = q_{L_R} \,. \quad (40)$$

If $L_1 = 1$, then trivially $\tilde{\Psi}_1 = \Psi_1$ and we can proceed to the degree q_2. Otherwise, we apply the above presented method to $\Psi_1, \dots, \Psi_{L_1}$ within the linear space $\mathcal{A}_1 = \mathbb{R}[\Psi_1, \dots, \Psi_{L_1}]_{\le q_{L_1}}$.

For the higher degrees, assume that for some index $1 < r \le R$ we have already constructed the sparse conservation laws $\tilde{\Psi}_1, \dots, \tilde{\Psi}_{L_{r-1}}$. We then choose a basis $p_1, \dots, p_{a_r}$ of the linear space $\mathcal{A}_r = \mathbb{R}[\tilde{\Psi}_1, \dots, \tilde{\Psi}_{L_{r-1}}, \Psi_{L_{r-1}+1}, \dots, \Psi_{L_r}]_{\le q_{L_r}}$. Since we started with functionally (and thus in particular algebraically) independent conservation laws, the $\mathbb{R}$-algebra generated by them is isomorphic to a polynomial ring in L_r variables and a linear basis of $\mathcal{A}_r$ is obtained by taking all "terms" $\tilde{\Psi}_1^{n_1} \cdots \tilde{\Psi}_{L_{r-1}}^{n_{L_{r-1}}} \cdots \Psi_{L_{r-1}+1}^{n_{L_{r-1}+1}} \cdots \Psi_{L_r}^{n_{L_r}}$ with integer exponents $n_i \ge 0$ satisfying $\sum_{i=1}^{L_r} n_i \le q_{L_r}$. For notational simplicity, we assume $p_i = \Psi_{L_{r-1}+i}$ for $1 \le i \le L_r - L_{r-1}$, but we do not care about the enumeration of the remaining polynomials in the basis.

For the construction of the first sparse conservation law of degree q_{L_r}, we augment (38) to the optimization problem

$$\begin{aligned} &\min_{\mathbf{a} \in \mathbb{R}^{L_r}} \quad \|C_r \mathbf{a}\|_1 \\ &\textbf{subject to} \quad \sum_{\ell=1}^{L} a_\ell = 1 \wedge \mathbf{e}_r^T \mathbf{a} > 0 \,. \end{aligned} \quad (41)$$

Here $\mathbf{e}_r \in \mathbb{R}^{L_r}$ has for the first $L_r - L_{r-1}$ entries a 1 and all remaining entries are 0. The matrix C_r has a_r columns representing the linear basis $p_1, \dots, p_{a_r}$ of $\mathcal{A}_r$. The condition $\mathbf{e}_r^T \mathbf{a} > 0$ ensures that the solution depends on at least one of the conservation laws $\Psi_{L_{r-1}+1}, \dots, \Psi_{L_r}$. Note that theoretically the optimal solution might satisfy $\mathbf{e}_r^T \mathbf{a} = 0$ and in this case would be missed by our ansatz. However, since our conservation laws are the result of a numerical process, this case is extremely unlikely and can be safely ignored.

If $L_r - L_{r-1} > 1$, we proceed again by orthogonality constraints to ensure the functional independence of the conservation laws of degree q_{L_r}. Note that we must enforce orthogonality only between the conservation laws of this degree, i. e. between the newly constructed conservation laws $\tilde{\Psi}_{L_{r-1}+1}, \dots, \tilde{\Psi}_{L_r}$. The previously constructed conservation laws $\tilde{\Psi}_1, \dots, \tilde{\Psi}_{L_{r-1}}$ are automatically functionally independent of the new ones for degree reasons.

Example 9. In our treatment of the FPUT lattice (Example 7), we assumed a priori knowledge of the linear conservation law $\Psi_1 = P$ given by (37). Furthermore, there are two linear conservation laws defined by the parameters $\Psi_2 = \alpha$ and $\Psi_3 = k$. We discovered an additional conservation law Ψ_4 of degree four (including the parameter dependency) which, however, was much more complicated than the Hamiltonian H given by (36). We now have to work in the linear space $\mathcal{A} = \mathbb{R}[\alpha, k, \Psi_1, \Psi_2]_{\le 4}$. A generating set of it is given by Ψ_2 and all "terms" $\alpha^a k^b \Psi_1^c$ with exponents $a, b, c \in \mathbb{N}_0$ satisfying $a + b + c \le 4$. A simple counting yields as dimension $\dim \mathcal{A} = 35$. Applying the above described approach followed by our refinement procedure, we can reduce the originally discovered fourth conservation law consisting of 135 terms to (36) with only one coefficient off by 2×10^{-7}.

V. SOME COMPLEXITY CONSIDERATIONS

The core computation in our approach is the kernel ridge regression requiring mainly the inversion of the matrix $K + \lambda \mathbb{1}$ (because of our indeterminate regression, we indeed need the inverse and not only the solution of a linear system). The dimension of this matrix equals the number $N_{\text{data}} = MN$ of used data points. It is a priori independent of the dimension D of the feature vector and of the degree q of the used polynomial kernel. In our experiments, typically N_{data} had to be roughly of the same magnitude as the dimension $n_{D,q}$ of our search space of polynomials in D variables and of degree up to q, but we expect that generally N_{data} grows slower than $n_{D,q}$. This implies in particular that modifications of our basic algorithm – like the "promotion" of parameters to dynamical variables or the inclusion of additional functions into the feature vector $\mathbf{x}$ which all increase D and may require higher values of q – have only a moderate effect on the complexity. Since our matrix is symmetric and positive definite, its inversion requires $\frac{1}{3} N_{\text{data}}^3 + O(N_{\text{data}}^2)$ operations. The (system) dimension D is relevant, when the data is produced by numerical integration. One may estimate the costs for this step to be roughly $O(D N_{\text{data}})$ which is, however, neglectable compared to the costs of the matrix inversion.

How many matrix inversions are needed depends on the grid search performed as part of the cross-validation: for every value of the regularization parameter λ that is tested, one inversion must be performed. This also implies that discovering more than one conservation law is cheap. Since we continue to use the same value of λ, each additional conservation law requires only one additional matrix inversion.

As soon as the final coefficient vector $\boldsymbol{\alpha}$ has been computed and the candidate conservation law Φ defined by (4) has been validated, we are in principle done and have found a conservation law (strictly speaking, already the labels $\mathbf{y}$ determine Φ via (6); we need $\boldsymbol{\alpha}$ mainly to

find $\mathbf{y}$!). Compared with approaches where conservation laws are represented by neural networks, our result is much more explicit, as it is a closed form expression (with numerical coefficients). However, as the conservation law is given as a linear combination of the dense polynomials $\varphi_{m,n}$, it does not arise in a form convenient for any subsequent analysis.

One should therefore transform the conservation law Φ obtained in the form (4) (or (6)) into the usual representation (11) of a polynomial. One may consider this rewriting as the analogue in our method to the symbolic regression step used in approaches based on neural networks to obtain a closed form expression for the discovered conservation law. If one does the expansion straightforwardly in a symbolic computation, then each of the base functions $\varphi_{m,n}(\mathbf{x}) = (\mathbf{x}^T\mathbf{x}_{m,n} + c)^q$ expands into a dense polynomial of degree q in D variables and we must compute and simplify a linear combination of N_{data} such polynomials each containing $n_{D,q}$ terms. Indeed, we noticed in experiments that – already for moderate values of D and q – such a symbolic computation leads to a bottleneck.

However, one can compute directly the coefficients of the representation (11) in a purely numerical manner. If we consider again the matrix S with entries $s_{mn,\boldsymbol{\mu}}$ defined by (31), then the coefficients of Φ arise as the product $S\boldsymbol{\alpha}$ and we obtain for the coefficients $a_{\boldsymbol{\mu}}$ in (11) the formula

$$a_{\boldsymbol{\mu}} = c^{\mu_0}\binom{q}{\hat{\boldsymbol{\mu}}}\sum_{m=1}^{M}\sum_{n=1}^{N}\alpha_{m,n}\mathbf{x}_{m,n}^{\boldsymbol{\mu}}\,, \tag{42}$$

where $\mu_0 = q - |\boldsymbol{\mu}|$. The numerical evaluation of this sum is straightforward. The total cost of the rewriting is then $O(n_{D,q}N_{\text{data}})$ and neglectable against the cost of the kernel ridge regression.

For finding the right trajectory labels $\mathbf{y}$, we must solve the linearly constrained quadratic program (9). It is known that such problems can be solved in polynomial time (roughly cubical) [30]. Note, however, that the program (9) is only M-dimensional and hence these costs are again neglectable against the cost of the kernel ridge regression. The same is true for our approach to discover multiple conservation laws, as (30) is still an M-dimensional program which only has more linear constraints. Since we always assume that $M > D$, the number of constraints is not relevant for the complexity.

Finally, we consider the costs of the sparsification process described above. The classical approach to linear programs like (39), the simplex algorithm and its variants, shows in rare cases an exponential worst-case behaviour, but the expected behaviour is a quadratic complexity in the number of decision variables (it has been proven that theoretically linear programs can always be solved in polynomial time with the fastest algorithms having the same time complexity as fast matrix multiplication, i. e. better than cubic) [31]. The number of decision variables can be rather large here, namely up to $L + n_{D,q}$. However, in practise, this number is much smaller: even if the conservation laws Ψ_ℓ are not optimal, we can expect that each of them contains much less than $n_{D,q}$ terms. In all our examples, this was indeed the case. Thus we can assume that also the sparsification is cheaper than the kernel ridge regression.

We thus conclude that our whole method has a complexity given by $\frac{1}{3}(N_\lambda + N_{CL})N_{\text{data}}^3 + O(N_{\text{data}}^2)$ where N_λ denotes the number of λ-values used in the grid search and N_{CL} the number of functionally independent conservation laws discovered. We note that this very basic estimate neglects the many available methods to speed up the inversions in a ridge regression ranging from fast updates via the Woodbury formula over aiming for optimal statistical accuracy to the use of hierarchical matrices. State of the art methods achieve complexities of $O(N_{\text{data}}\sqrt{N_{\text{data}}})$ and better [32, 33] and allow for the handling of millions of data points.

For a rough comparison with neural networks, let us assume that we have a simple feedforward network with N_{layer} layers each containing D neurons. We further assume that the backpropagation requires N_{iter} iterations. Then it is well-known that the complexity for processing one data point is $O(N_{\text{iter}}N_{\text{layer}}D^3)$. We denote by N_{ep} the number of epochs used for the training of the neural network. By a rule of thumb, N_{ep} grows with N_{data} and we assume for simplicity a linear growth, i. e. $N_{\text{ep}} = O(N_{\text{data}})$. Since in each epoch every data point has to be run through the network, we thus obtain a complexity estimate of $O(N_{\text{data}}^2)$ with a leading coefficient of the size $O(N_{\text{iter}}N_{\text{layer}}D^3)$.

If we contrast this estimate with our approach (without fast inversions), one must take into account that not only our leading coefficient is much smaller, but that in particular our approach requires much less data points than neural networks. If we compare with examples presented in the literature, then our N_{data} is typically at least an order of magnitude smaller than the N_{data} used there. Thus only for very large examples neural networks might be cheaper and even this is doubtful, if one uses state of the art inversion algorithms. This comparison has not yet taken into account that our approach yields directly a nice symbolic representation of the conservation laws, whereas approaches based on neural networks still need a symbolic regression step to translate the network into a formula. The cost of this step is difficult to estimate and depends of course on the used symbolic regression method, but it is surely not neglectable.

One may wonder why we bother with machine learning techniques, if we search only for polynomial conservation laws – in particular, in view of our refinement method. One could instead directly make a simple ansatz for a general polynomial of degree q in D variables with indeterminate coefficients, enter the ansatz into the partial differential equation (2) and then solve the arising linear system of dimension for its $n_{D,q}$ coefficients.

For an exact computation, one must then assume that the right hand side of the dynamical system (1) consists at most of rational functions over the rational num-

bers $\mathbb{Q}$, as only then entering a polynomial ansatz yields after clearing denominators a linear system over $\mathbb{Q}$. The usual complexity estimates for nullspace computations assumes that any arithmetical operation can be performed in constant time. While this is surely the case for floating point operations, it is not true for computations over the rationals $\mathbb{Q}$. A naive implementation of Gaussian implementation will even have an exponential bit complexity, as the number of digits will double in each elimination step. With more advanced methods like modular arithmetics, one can again obtain algorithms with a cubic complexity. However, the constants are much worse than in a floating point computation and depend on the size of the results. As a rule of thumb, one can treat on a single processor machine linear systems over $\mathbb{Q}$ up to a size of about a few thousand indeterminates within reasonable computation times.

If one replaces the exact nullspace computation of the linear system by a numerical one, then one finds the classical complexity $O(n_{D,q}^3)$. Assuming that MN and n_{D_q} are of the same order of magnitude (which is roughly the case in all examples that we studied), we find the same complexity as in our approach. A key difference is that in our approach it is not necessary to know the dynamical system (1) explicitly; it suffices, if enough trajectory data are available. If (1) is explicitly given, then our refinement step takes into account only those power products which according to the regression analysis actually appear in the conservation law. Thus generally in our refinement step one has to solve a much smaller linear system and its costs are again neglectable.

In principle, one can also work with an ansatz containing more general functions than polynomials. The "theorist" approach of [6] is an example how this can be realized purely numerically leading to a linear regression problem. For exact computations, their approach is not suitable, as it would lead to a linear system over a function field instead of the rationals $\mathbb{Q}$. For doing exact linear algebra computations over such a field, one must be able to decide whether or not an expression is zero. Richardson's theorem (see [34, Sect. 5]) asserts that this problem becomes rapidly undecidable, if one adds transcendental functions like $\exp x$ or $\sin x$.

However, with a suitable setup we can always reduce to linear algebra over the rationals. We first choose a (finite-dimensional) $\mathbb{Q}$-linear function space V_1 which represents the search space in which we look for conservation laws. Then we have to construct a second (finite-dimensional) $\mathbb{Q}$-linear function space V_2 which must contain for each function $\Phi \in V_1$ and for each variable x_d the product $(\partial\Phi/\partial x_d)f_d$. Let $\{g_1, \ldots, g_R\}$ be a linear basis of V_1 and $\{h_1, \ldots, h_S\}$ one of V_2. Making the ansatz $\Phi = \sum_{r=1}^{R} \beta_r g_r$ with yet undetermined coefficients $\beta_r \in \mathbb{Q}$ and entering it into the partial differential equation (2), we obtain a condition of the form $\sum_{s=1}^{S} c_s(\boldsymbol{\beta}) h_s = 0$ where the coefficients c_s are linear in the unknowns $\boldsymbol{\beta}$. The ansatz defines an exact conservation law, if and only if $c_s(\boldsymbol{\beta}) = 0$ for $s = 1, \ldots, S$. This condition defines a linear system over $\mathbb{Q}$ for the coefficients $\boldsymbol{\beta}$. Compared with the purely polynomial case, the dimensions R and S are rapidly growing here for more complicated situations, so that exact computations will often be unfeasable.

VI. FURTHER APPLICATIONS

The basic idea of our approach is fairly independent of the notion of a conservation law of a dynamical system. It is concerned with predicting functions from knowing points in their level sets (the "grouped data" of Ha and Jeong [3]). This problem also appears in other situations. We briefly discuss here three applications. The first one is very close to what we have done so far: we consider *discrete* dynamical systems which can be handled by our approach without any real changes. As a second application, we consider discovering simultaneous conservation laws of several vector fields which is equivalent to finding an implicit representation of the integral manifolds of an (integrable) vector field distribution. The final application is quite different: the implicitization of curves or higher-dimensional surfaces which is important in algebraic geometry, geometric modeling and computer vision.

A. Discrete Dynamical Systems

Our approach can also be applied to *discrete* dynamical systems, i. e. systems where the time t is a discrete variable. We assume $t \in \mathbb{Z}$ and consider an autonomous first-order system of the form

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t)\,. \tag{43}$$

A *conservation law* of it is a function $\Phi\colon \mathbb{R}^D \to \mathbb{R}$ which remains constant along solutions of (43), i. e. which satisfies $\Phi(\mathbf{x}_{t+1}) = \Phi(\mathbf{x}_t)$ (see e. g. [35]).

Thus we are in exactly the same situation, as in the continuous case: we know points on level sets of Φ and can apply the approach developed above. One may even say that the discrete case is slightly easier to handle, as evaluation of (43) along a trajectory does not require the use of numerical approximations as in the case of differential equations. On the other hand, it is more difficult to provide a more or less uniform sampling of the state space, as one has no control about the distance between consecutive points.

Remark 10. In the differential case, it was not really necessary to assume that we treat a first-order system. Most numerical methods are geared towards such systems, but e. g. there also exist methods for second-order systems. For our approach this is irrelevant; we only need the trajectory data (this is different for the approach by Liu et al. [5] using the partial differential equation (2) which assumes a first-order system). In the discrete case, the above definition of a conservation law is valid only for

first-order systems. For a system of order Q, conservation laws are functions defined on $\mathbb{R}^{QD}$, as they depend not only on $\mathbf{x}_t$, but also on the points $\mathbf{x}_{t+1}, \dots, \mathbf{x}_{t+Q-1}$, i. e. on a whole segment of the trajectory [35]. In principle, our approach can be adapted to such a situation, but it is probably easier to rewrite the given dynamical system as a first-order one.

Example 11. We consider the scalar second-order difference equation

$$x_{t+2} = \frac{t}{t+1} x_t + \frac{1}{x_{t+1}} \tag{44}$$

appearing in [35]. One readily verifies that

$$\Phi(t, x_t, x_{t+1}) = t x_t x_{t+1} - \frac{1}{2} t(t+1) \tag{45}$$

is constant along solutions and thus defines a polynomial conservation law. For applying our approach to this equation, we first rewrite it as a first-order system by introducing $y_t = x_{t+1}$ and then render it autonomous by introducing $z_t = t$. This yields the rational system

$$x_{t+1} = y_t\,, \quad y_{t+1} = \frac{x_t z_t}{z_t + 1} + \frac{1}{y_t}\,, \quad z_{t+1} = z_t + 1 \tag{46}$$

with a polynomial conservation law of degree 3

$$\Phi(x_t, y_t, z_t) = x_t y_t z_t - \frac{1}{2} z_t(z_t + 1)\,. \tag{47}$$

Applying our approach to (46) and rounding the obtained coefficients to four digits, we discover the exact conservation law (47).

Remark 12. As the approach of Arora et al. [7] is based on finding via discrete gradients a particularly well adapted discretization of the given continuous dynamical system, it cannot be extended to discrete dynamical systems. Similarly, the approach of Kaiser et al. [1] uses the Koopman theory of continuous dynamical systems and cannot be extended. The neural deflation approach of Zhu et al. [9] only applies to Hamiltonian systems and requires a Poisson structure. While there exist discrete analogues of these concepts, it is unclear whether an extension is possible. The approach of Liu et al. [5] is based on learning solutions of the partial differential equation (2). For discrete dynamical systems, one can derive a discrete analogon to this partial differential equation where partial derivatives are replaced by shift operators [35]. It is unclear how solutions of this difference equation can be learned, but our refinement procedure could probably be adapted to this partial difference equation. By contrast, the approaches of Ha and Jeong [3] via a noise-variance loss and Wetzel et al. [2] via Siamese neural networks, respectively, can also be straightforwardly extended to the discrete case.

B. Integral Manifolds of Vector Field Distributions

Let the vector fields $X_1, \dots, X_A$ span a smooth and regular distribution $\mathcal{A}$ on $\mathbb{R}^D$ of rank A, i. e. at each point $\mathbf{x} \in \mathbb{R}^D$ the vectors $X_1(\mathbf{x}), \dots, X_A(\mathbf{x})$ are linearly independent. An *integral manifold* of $\mathcal{A}$ is an A-dimensional submanifold $\mathcal{M} \subseteq \mathbb{R}^D$ such that at each point $\mathbf{m} \in \mathcal{M}$ the vectors $X_1(\mathbf{m}), \dots, X_A(\mathbf{m})$ form a basis of the tangent space $T_{\mathbf{m}}\mathcal{M}$. The distribution $\mathcal{A}$ is called *integrable*, if there exists an integral manifold through every point $\mathbf{x} \in \mathbb{R}^D$, and *involutive*, if the Lie bracket of any two vector fields X_a, $X_{a'}$ also lies in $\mathcal{A}$, i. e. if there exist smooth functions $C_{aa'}^{a''}(\mathbf{x})$ such that $[X_a, X_{a'}] = \sum_{a''=1}^{A} C_{aa'}^{a''} X_{a''}$. By the well-known Frobenius theorem, a smooth and regular distribution is integrable, if and only if it is involutive [36, Chapt. 19]. The integral manifolds of an integrable distribution define a *foliation* of $\mathbb{R}^D$.

In the sequel, we will always assume that $\mathcal{A}$ is an involutive and hence also integrable distribution spanned by explicitly given vector fields $X_1, \dots, X_A$. Our goal is to discover an *implicit representation* of the foliation defined by $\mathcal{A}$. This means that we search for $B = D - A$ functionally independent functions $\Phi_b(\mathbf{x})$ such that each integral manifold can be represented as the solution set of the system $\Phi_b(\mathbf{x}) = c_b$ with $b = 1, \dots, B$ for suitable constants $c_b \in \mathbb{R}$.

For a dynamical system defined by a single vector field X, the linear first-order partial differential equation (2) characterizing conservation laws can be written as $X\Phi = 0$. We now generalize it to an overdetermined linear first-order *system* $X_a\Phi = 0$ with $a = 1, \dots, A$ and search for a linear basis of the solution space.

It follows immediately from the definitions that each of the functions $\Phi_b(\mathbf{x})$ must be a conservation law of each of the vector fields X_a. Thus we search for functions which are simultaneously conservation laws for several vector fields. Within our approach, this is easy to handle. We choose some random initial points $\mathbf{x}_0^{(m)}$ with $m = 1, \dots, M$. Then we numerically compute N points $\mathbf{x}_{m,n}^{(a)}$ on the trajectory of each vector field X_a through each initial point $\mathbf{x}_0^{(m)}$. All points with the same index m must lie on the same level set of the functions Φ_b. Hence we can use exactly the same indeterminate regression approach, but now with larger groups of data: each group contains the data of A trajectories – one for each vector field X_a. Note that this time we know already in advance the number of functionally independent conservation laws to be discovered, namely B.

Remark 13. Many textbooks present approaches for computing integral manifolds where it is assumed that the vector fields X_a commute with each other, i. e. that all functions $C_{aa'}^{a''}$ vanish. Our approach does not require such assumptions; we only need sufficiently many points lying on different level sets.

Example 14. We consider on $\mathbb{R}^4$ the two vector fields

$$X_1 = x_1\partial_{x_2} - x_2\partial_{x_1} + x_3\partial_{x_4} - x_4\partial_{x_3}\,,$$
$$X_2 = x_3\partial_{x_1} - x_1\partial_{x_3} + x_4\partial_{x_2} - x_2\partial_{x_4}\,. \qquad (48)$$

As their Lie bracket vanishes, they span an involutive distribution $\mathcal{A}$. It is easy to verify that the integral manifolds can be described by the two quadratic functions

$$\Phi_1(\mathbf{x}) = x_1^2+x_2^2+x_3^2+x_4^2\,, \quad \Phi_2(\mathbf{x}) = x_1x_4-x_2x_3\,. \quad (49)$$

Applying our approach for the data condition $(M, N) = (5, 20)$, we discover first the conservation law

$$\begin{aligned}\Phi = &-0.008836x_1^2 - 0.00889x_2^2\\ &- 0.008794x_3^2 - 0.008838x_4^2\\ &- 0.011600x_2x_3 + 0.011605x_1x_4\end{aligned} \qquad (50)$$

which obviously corresponds with high accuracy to a linear combination of Φ_1 and Φ_2. The generalization error was $\epsilon_{\text{gen}} = 7.8 \times 10^{-6}$ and the validation error $\Delta_1(\Phi) = 1.26 \times 10^{-5}$.

Searching for a further conservation law orthogonal to the first one, we discover in a second run

$$\begin{aligned}\tilde{\Phi} = &\,0.0037843x_1^2 + 0.0037844x_2^2\\ &+ 0.00378439x_3^2 + 0.00378439x_4^2\\ &- 0.0135777x_2x_3 + 0.013577x_1x_4\end{aligned} \qquad (51)$$

approximating very well another linear combination of Φ_1 and Φ_2. This time, we have a generalization error of $\epsilon_{\text{gen}} = 1.72 \times 10^{-6}$ and a validation error of $\Delta_1(\tilde{\Phi}) = 5.62\times10^{-6}$. Calling our sparsification procedure with the two discovered expressions Φ and $\tilde{\Phi}$, we obtain with very high precision Φ_1 and Φ_2. Again this is due to the fact that these two conservation laws are indeed orthogonal to each other.

Remark 15. The problem treated here concerns discovering an implicit representation of a family of A-dimensional manifolds. Naively, one would expect that the number of data points needed should grow exponentially with A, as so many points are required for a proper sampling of the manifolds (the often mentioned "curse of dimensionality"). However, this is not the case with our approach. Here the number of data points grows only linearly with A, since A trajectories are computed for every chosen initial point $\mathbf{x}_0^{(m)}$. As already mentioned earlier, we typically choose M initial points where M should be larger than the dimension D of the ambient space, but is typically of the same order of magnitude. Thus the total number of data points is $O(AD)$. Again, this is due to the fact that we are not using random points, but grouped data with points lying on special curves, namely trajectories of the given vector fields.

C. Implicitization of Curves and Surfaces

In geometry, two different types of representations of objects like curves and surfaces (of dimension 2 or higher) in some affine space $\mathbb{R}^D$ are typically used. In an *explicit* representation, a K-dimensional object is given via a parametrization $x_d = \varphi_d(t_1, \ldots, t_K)$ with parameters $(t_1, \ldots, t_K) \in P \subseteq \mathbb{R}^K$ from some real parameter set. In an *implicit* representation, the object is described as the zero set of some functions $\Phi_j \colon \mathbb{R}^D \to \mathbb{R}$. Each representation is preferable for certain tasks. For example, an explicit representation allows us to produce easily points on the object, whereas an implicit representation is better for checking whether a given point lies on the object. It is therefore of importance to be able to switch between these two types of representations and implicitization is the problem of going from a given parametrization to an implicit representation. An extensive discussion in the context of geometric modeling can be found in [37].

In algebraic geometry, the given parametrization φ_d is typically rational and one looks for polynomials Φ_j, i. e. it is assumed that the represented object is a variety. Then implicitization can be formulated as an elimination problem and techniques like resultants or Gröbner bases are available for its solution (see [37] and references therein). However, these techniques are expensive and often lead to polynomials of high degree making subsequent computations costly and numerically unstable (which is a problem for applications in CAGD).

Consequently, numerous methods for approximate implicitization have been developed (see the references in the recent work [38]) and our approach is related to a class of methods called *discrete approximate implicitization*. The authors of [39] propose to use an autoencoder for fitting polynomial equations, i. e. a neural network. Like for conservation laws, we consider our approach via kernel methods as much cheaper.

While many of the algebraic methods can – at least in principle – handle arbitrary dimensions $K < D$, most of the approximate approaches concentrate on the case $K = D-1$ for $D = 2, 3$, i. e. on planar curves and surfaces in 3D, as these are the dominant situations in CAGD and geometric modelling. This restriction means that the implicit representation consists of a single polynomial. For simplicity, we also consider here only this case.

Our discovery of conservation laws of ordinary differential equations from trajectory data corresponds to the implicitization of whole *families* of curves: we are looking for an implicit representation valid not only for a single curve, but for many curves simultaneously. This means that we search for functions $\mathbf{\Phi}$ such that the kth curve is described by the equations $\mathbf{\Phi}(\mathbf{x}) = \mathbf{c}^{(k)}$ for some constant vector $\mathbf{c}^{(k)}$. The extension of our approach to families of two or higher dimensional surfaces is straightforward: one only needs a way to generate sufficiently many data points on the surfaces that do not lie on lower-dimensional subsets. While for a single curve or surface always an implicitization exists (though not necessarily one with polynomials as desired in algebraic geometry), this represents a special property for families.

In the case of a single curve or surface, one needs in addition to data points on the curve or surface one point

off the curve or surface; this point should not lie too close. Then we take as labels for all points on the curve or surface 0 and for the one additional point 1.

Example 16. A classical planar algebraic curve is the *trifolium* shown in Figure 4. A trigonometric parametrization of it is given by

$$\begin{aligned} x &= 4\sin(t)^4 - 3\sin(t)^2\,, \\ y &= -\sin(t)\cos(t)(4\sin(t)^2 - 3) \end{aligned} \tag{52}$$

with $0 \le t \le \pi$. As an implicit description of it, one may use the polynomial equation of degree 4

$$(x^2+y^2)^2 = x(x^2-3y^2)\,. \tag{53}$$

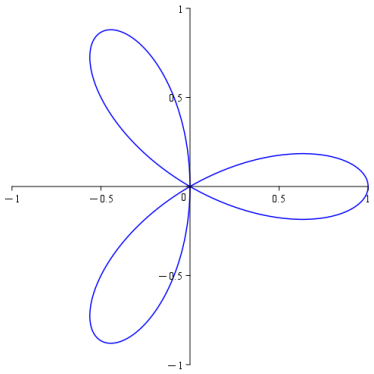


FIG. 4: Regular trifolium.

A point outside of the curve is for example $(-1, 0)$. By employing our kernel method, we discover – after rounding the coefficients to five digits – the exact solution $x^4 - x^3 + 2x^2y^2 + 3xy^2 + y^4 = (x^2+y^2)^2 - x(x^2-3y^2)$.

Example 17. *Whitney's umbrella* is the algebraic surface shown in Figure 5 – note that the "handle", i. e. the z-axis, is part of the surface. It is implicitly defined by the polynomial equation of degree 3

$$x^2 z = y^2\,. \tag{54}$$

A polynomial parametrization of the two-dimensional part of the surface is given by

$$x = u\,, \quad y = uv\,, \quad z = v^2\,. \tag{55}$$

Points on the z-axis are not really necessary for the implicitization; it suffices to consider only points on the two-dimensional part of the surface. A point obviously off Whitney's umbrella is $(1, 1, -1)$. By employing a polynomial kernel of degree 3, we discover after rounding the coefficients to 7 digits the exact implicit description.

VII. CONCLUSIONS

In this work, we proposed a novel approach to discover either a single conservation law of a dynamical system or

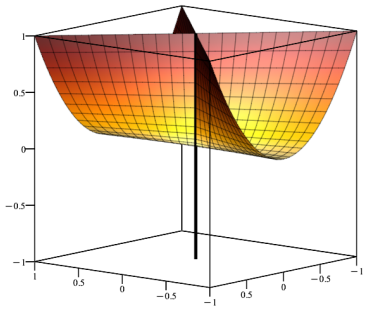


FIG. 5: Whitney's umbrella.

a complete set of functionally independent conservation laws. It is based on a kernel method, namely an "indeterminate" ridge regression, instead of some variants of neural networks. We believe that it is computationally more efficient and in particular requires much less data which is important, if the data comes from experiments and not from numerical simulations. Indeed, in all our examples a few dozens to a few hundred data points were sufficient (600 points was the maximum we used). By contrast, in [2, 7, 9], the authors report that for problems of similar sizes they trained their neural networks with $50,000$ to $200,000$ points; only the $2,000$ data points used in [3] are somewhat close to our values. The neural networks typically consist of about 200 to $2,000$ neurons in the hidden layers and the training requires between $1,000$ and $50,000$ epochs. The computational costs of such trainings are probably several orders of magnitudes larger than the matrix inversions needed in a kernel ridge regression, even if one takes into account that the determination of the labels in our indeterminate regression and the tuning of the regularization parameter require repeated inversions. The comparatively low number of data points ensures that the costs for the inversions remain moderate. We also emphasize that our approach yields immediately a symbolic representation of a discovered conservation law, whereas methods based on neural networks still need a subsequent symbolic regression step which usually also comes with considerable computational costs.

For discovering conservation laws, one may distinguish at least three different scenarios. In the first and simplest one – on which we concentrated in this work – the dynamical system (1) is explicitly known. Thus arbitrary amounts of synthetic trajectory data can be produced any time. For most dynamical systems, the cost of the required numerical integrations is negligible compared to the subsequent machine learning and the obtained data are fairly accurate. A sampling bias can be easily avoided by using a normalized vector field. Furthermore, one can use the explicitly given vector field for a symbolic refinement of candidate conservation laws. Finally, the explicit knowledge of the system typically indicates natural choices for an extended feature vector to proceed beyond

polynomial conservation laws.

In a second scenario, the dynamical system is only known as a black box (e. g. in form of a neural network). Then one can still produce arbitrary amounts of accurate synthetic trajectory data, but it becomes more difficult to avoid a sampling bias (e. g. via interpolation). A symbolic refinement is no longer possible and extensions of the feature vector can only be guessed.

In a third scenario, the dynamical system is not known at all and only a limited and fixed amount of possibly noisy data is available. Such a situation arises in particular when one has experimental data without a mathematical model and raises a number of questions that call for a deeper analysis. An obvious one is the robustness with respect to noisy data. One could hope that up to a certain noise level the regularization parameter of the kernel ridge regression may compensate the noise. Another question is the robustness against the sampling bias introduced by unevenly distributed data. Here we are optimistic that the effect is not very pronounced. In both cases, only further experiments will tell whether our expectations are justified.

Finally, we want to mention that our approach should also be able to discover *approximate* conservation laws. These could be for example quantities that are almost conserved on shorter time scales, but show some evolution on longer time scales or quantities that remain conserved only after an initial transient phase. While in physics many models are constructed in such a way to possess exact symmetries and thus related exact conservation laws, we expect that in the phenomenological models that prevail in biology approximate conservation laws are much more frequent than exact ones. In our approach, one can study such phenomena by data windowing. Thus one could e. g. consider only trajectory points up to a certain maximal time or conversely only points at times larger than a certain minimal time. By playing with the time threshold, one may even glean information about the corresponding time scales.

DATA AVAILABILITY

A repository with the Python code and the data used for the examples presented in this article is publicly available on Zenodo at the DOI `https://doi.org/10.5281/zenodo.14254092`.

ACKNOWLEDGMENTS

The work of MAM and WMS was performed within the Research Training Group *Biological Clocks on Multiple Time Scales* (GRK 2749) at Kassel University funded by the German Research Foundation (DFG). We are indebted to Rémi Gribonval for raising our attention to the problem studies in Section VI B and pointing us to [40] where it arises in the context of analysing the training of neural networks.

[1] E. Kaiser, J. Kutz, and S. Brunton, Discovering conservation laws from data for control, in *2018 IEEE Conference on Decision and Control* (IEEE, 2018) pp. 6415–6421.
[2] S. Wetzel, R. Melko, J. Scott, M. Panju, and V. Ganesh, Discovering symmetry invariants and conserved quantities by interpreting Siamese neural networks, Phys. Rev. Res. **2**, 033499 (2020).
[3] S. Ha and H. Jeong, Discovering conservation laws from trajectories via machine learning, Preprint arXiv:2102.04008 (2021).
[4] Z. Liu and M. Tegmark, Machine learning conservation laws from trajectories, Phys. Rev. Lett. **126**, 180604 (2021).
[5] Z. Liu, V. Madhavan, and M. Tegmark, Machine learning conservation laws from differential equations, Phys. Rev. E **106**, 045307 (2022).
[6] Z. Liu, P. Sturm, S. Bharadwaj, S. Silva, and M. Tegmark, Interpretable conservation laws as sparse invariants, Phys. Rev. E **109**, L023301 (2024).
[7] S. Arora, A. Bihlo, R. Brecht, and P. Holba, Model-agnostic machine learning of conservation laws from data, Preprint arXiv:2301.07503 (2023).
[8] P. Lu, R. Dangovski, and M. Soljačić, Discovering conservation laws using optimal transport and manifold learning, Nature Comm. **14**, 4744 (2023).
[9] W. Zhu, H. Zhang, and P. Kevrekidis, Machine learning of independent conservation laws through neural deflation, Phys. Rev. E **108**, L022301 (2023).
[10] A. Belyshev, A. Kovrigin, and A. Ustyuzhanin, Beyond dynamics: Learning to discover conservation principles, Mach. Learn. Sci. Tech. **5**, 025055 (2024).
[11] S. Kung, *Kernel Methods and Machine Learning* (Cambridge University Press, Cambridge, 2014).
[12] B. Schölkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond* (MIT Press, Boston, 2018).
[13] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis* (Cambridge University Press, New York, 2004).
[14] P. Olver, *Applications of Lie Groups to Differential Equations*, Graduate Texts in Mathematics 107 (Springer-Verlag, New York, 1986).
[15] G. Bluman and S. Anco, *Symmetry and Integration Methods for Differential Equations*, Applied Mathematical Sciences 154 (Springer-Verlag, New York, 2002).
[16] X. Zhang, *Integrability of Dynamical Systems: Algebra and Analysis*, Developments in Mathematics (Springer Nature, Singapore, 2018).
[17] W. Miller Jr, S. Post, and P. Winternitz, Classical and quantum superintegrability with applications, J. Phys. A: Math. Theor. **46**, 423001 (2013).

[18] R. Schimming, Conservation laws for Lotka–Volterra models, Math. Meth. Appl. Sci. **26**, 1517 (2003).

[19] D. Angeli, A tutorial on chemical reaction network dynamics, Eur. J. Control **15**, 398 (2009).

[20] T. Dauxois and M. Peyrard, *Physics of Solitons* (Cambridge University Press, New York, 2006).

[21] S. Englander, N. Kallenbach, A. Heeger, J. A. Krumhansl, and S. Litwint, Nature of the open state in long polynucleotide double helices: Possibility of soliton excitations, Proc. Natl. Acad. Sci. **77**, 7222 (1980).

[22] M. Feinberg, *Foundations of Chemical Reaction Network Theory*, Applied Mathematical Sciences 202 (Springer Nature, Cham, 2019).

[23] H. Errami, M. Eiswirth, D. Grigoriev, W. Seiler, T. Sturm, and A. Weber, Detection of Hopf bifurcations in chemical reaction networks using convex coordinates, J. Comput. Phys. **291**, 279 (2015).

[24] E. Fermi, J. Pasta, and S. Ulam, Studies of non linear problems, Los Alamos Report LA-1940 (1955).

[25] M. Toda, Vibration of a chain with nonlinear interaction, J. Phys. Soc. Jpn. **22**, 431 (1967).

[26] M. Toda, Waves in nonlinear lattice, Prog. Theor. Phys. Suppl. **45**, 174 (1970).

[27] T. Coleman and A. Pothen, The null space problem I. Complexity, SIAM J. Alg. Discr. Meth. **7**, 527 (1986).

[28] E. Candès, J. Romberg, and T. Tao, Stable signal recovery from incomplete and inaccurate measurements, Commun. Pure Appl. Math. **59**, 1207 (2006).

[29] D. Donoho, For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution, Commun. Pure Appl. Math. **59**, 797 (2006).

[30] S. Vavasis, Complexity theory: Quadratic programming, in *Encyclopedia of Optimization*, edited by C. Floudas and P. Pardalos (Springer-Verlag, Boston, 2001) pp. 300–304.

[31] N. Megiddo, On the complexity of linear programming, in *Advances in Economic Theory*, edited by T. Bewley (Cambridge University Press, 1987) pp. 225–268.

[32] G. Chávez, Y. Liu, P. Ghysels, X. Li, and E. Rebrova, Scalable and memory-efficient kernel ridge regression, in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (2020) pp. 956–965.

[33] A. Rudi, L. Carratino, and L. Rosasco, FALKON: An optimal large scale kernel method, in *Advances in Neural Information Processing Systems 20*, edited by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (2017) pp. 3888–3898.

[34] B. Buchberger and R. Loos, Algebraic simplification, in *Computer Algebra: Symbolic and Algebraic Computation*, Computing Supplementum 4, edited by B. Buchberger, G. Collins, R. Loos, and R. Albrecht (Springer-Verlag, Wien, 1982) pp. 11–43.

[35] P. Hydon, Symmetries and first integrals of ordinary difference equations, Proc. Roy. Soc. A **456**, 2835 (2000).

[36] J. Lee, *Introduction to Smooth Manifolds*, Graduate Texts in Mathematics 218 (Springer-Verlag, New York, 2003).

[37] C. Hoffmann, *Geometric and Solid Modeling: An Introduction* (Morgan Kaufmann, San Mateo, 1989).

[38] M. Guo and Y. Gao, Adaptive approximate implicitization of planar parametric curves via asymmetric gradient constraints, Symmetry **15**, 1738 (2023).

[39] G. Wang, W. Li, L. Zhang, L. Sun, P. Chen, L. Yu, and X. Ning, Encoder-X: Solving unknown coefficients automatically in polynomial fitting by using an autoencoder, IEEE Trans. Neural Netw. Learn. Sys. **33**, 3264 (2022).

[40] S. Marcotte, R. Gribonval, and G. Peyré, Abide by the law and follow the flow: Conservation laws for gradient flows, Preprint arXiv 2307.00144 (2023).