Uncovering differential equations from data with hidden variables

Agustín Somacal[*] and Yamila Barrera[†]
Aristas S.R.L., Dorrego 1940, Torre A, 2do Piso, dpto. N (1425), Ciudad Autónoma de Buenos Aires, Argentina

Leonardo Boechi[‡] and Matthieu Jonckheere[§]
Instituto de Calculo-CONICET, Intendente Güiraldes 2160, Ciudad Universitaria, Pabellón II, 2do. piso, (C1428EGA), Buenos Aires, Argentina

Vincent Lefieux[‖]
Réseau de Transport d'Electricité (RTE), 92060 Paris, France

Dominique Picard[¶]
Université de Paris, LPSM, UFR Mathematiques Batiment Sophie Germain, 75013 Paris, France

Ezequiel Smucler[#]
Universidad Torcuato Di Tella, Av. Figueroa Alcorta 7350 (C1428BCW) Sáenz Valiente 1010 (C1428BIJ) Ciudad de Buenos Aires, Argentina and Aristas S.R.L., Dorrego 1940, Torre A, 2do Piso, dpto. N (1425), Ciudad Autónoma de Buenos Aires, Argentina

(Received 3 June 2021; accepted 12 April 2022; published 20 May 2022)

SINDy is a method for learning system of differential equations from data by solving a sparse linear regression optimization problem [Brunton, Proctor, and Kutz, Proc. Natl. Acad. Sci. USA **113**, 3932 (2016)]. In this article, we propose an extension of the SINDy method that learns systems of differential equations in cases where some of the variables are not observed. Our extension is based on regressing a higher order time derivative of a target variable onto a dictionary of functions that includes lower order time derivatives of the target variable. We evaluate our method by measuring the prediction accuracy of the learned dynamical systems on synthetic data and on a real data set of temperature time series provided by the Réseau de Transport d'Électricité. Our method provides high quality short-term forecasts and it is orders of magnitude faster than competing methods for learning differential equations with latent variables.

DOI: 10.1103/PhysRevE.105.054209

I. INTRODUCTION

Many branches of science are based on the study of dynamical systems. Examples include meteorology, biology, and physics. The usual way to model deterministic dynamical systems is by using (partial) differential equations. Typically, differential equation models for a given dynamical system are derived using *a priori* insights into the problem at hand; then the model is validated using empirical observations. In an era in which massive data sets pertaining to different fields of science are widely available, an interesting problem is whether it is possible for a useful differential equations model to be learned directly from data, without any major modeling effort required by the researcher.

The SINDy method is an approach of sparse identification of nonlinear dynamical systems that consists of linking the dynamical system discovery problem to a statistical regression problem [1–3]. The main idea of the SINDy method is to consider a set of differential operators (possibly in both time and space if appropriate), discretize them, for example by using finite differences, and then regress the outcome of interest on the discretized differential operators. By solving the regression problem using an *ad hoc* thresholded least-squares algorithm, they can build sparse, interpretable models, that use mostly low order derivatives. The authors explored the applicability of their method on simulated data, but only in situations in which all the variables of the simulated models are observed.

Our goal is to extend the SINDy model for the case in which not all relevant variables are observed, that is, in cases in which the main variable of interest depends on other variables of which no measurements are available. As an example of application we consider the climate time series of the Réseau de Transport d'Électricité (RTE). RTE is the main electricity network operational manager in France, who is interested in understanding the behavior of climate time series because of their impact on energy consumption. RTE

[*]a.somacal@aristas.com.ar
[†]y.barrera@aristas.com.ar
[‡]lboechi@ic.fcen.uba.ar
[§]mjonckhe@dm.uba.ar
[‖]vincent.lefieux@rte-france.com
[¶]picard@math.univ-paris-diderot.fr
[#]e.smucler@aristas.com.ar

uses high-level simulations of hourly temperature series to study the impact different climate scenarios have on electricity consumption, and hence on the French electrical power grid. The underlying simulations are based on the Navier-Stokes equations and include variables as wind velocity, density, pressure, etc. The resulting dynamic system is known to be chaotic, see [4]. For that reason, our goal is to learn a system of differential equations that adequately models the dynamics of the temperature time series if the only observable variable is temperature, that is, if pressure, wind velocity, etc., are hidden variables.

To accommodate the possibility of hidden variables we note that, for a large class of dynamical systems, it is possible to reconstruct a trajectory (equivalent to the original one) given only one of the model variables, using its higher order derivatives [5,6].

Related to our approach is the Generalized Polynomial Modeling method (known as GPoMo), that addresses the recovery problem via a combinatorial search among a predefined set of polynomial functions of the observable variables [7] and [8]. The GPoMo method proceeds by choosing iteratively a family of combination of terms that minimize the Akaike or the Bayesian information criterion. Finally, it returns the set of best models. The authors also discuss the ability of their algorithm to find equations able to capture the dynamics when only some variables are observed. However, we will show that unfortunately this approach does not scale to large problems. Other approaches for learning dynamical systems from data available in the literature, such as those based on symbolic regression [8], also have the drawback of being too computationally expensive.

The article is organized as follows. In Sec. II we review the GPoMo and SINDy methods and afterwards, describe our methodology (named L-ODEfind) in detail. Section III presents the results of our experiments. In particular, in Sec. III B, we compare the performance of GPoMo and L-ODEfind in recovering differential equations using empirical data in the case in which all relevant variables are observed. In Sec. III C we compare them in the harder case in which at least one relevant variable driving the dynamical system is latent. We apply our proposed method to real world temperature times series in Sec. III D. Finally, in Sec. IV we discuss future work and possible extensions.

II. METHODS

A. GPoMo: Differential equations recovery as a combinatorial search problem

GPoMo is a method proposed and implemented by [7] that addresses the differential equations recovery problem via a combinatorial search in the space of differential equations that can be expressed as polynomial functions of the observed variables. The method uses a genetic algorithm in which at each step new test models are generated by randomly choosing some polynomial terms to be included in the equations. This choice is made by making small variations (take or add a few terms) over the best previously seen models. Then, to select the winning models at each step, they are integrated and compared to the original data.

This combination of combinatorial search and integration steps makes the method slow (as we will see in the experiments). Moreover, except for polynomial combinations of the variables, it does not allow other types of regressors to be included, such as functions of the time variable. To overcome the aforementioned limitations of GPoMo, we based our approach on SINDy, which uses sparse regression to discover governing physical equations from measurement data. We briefly review the SINDy method in the following section.

B. SINDy: Differential equations recovery as a linear regression problem

In our work we focus on ordinary differential equations even though SINDy can also tackle partial differential equations. In particular, consider a dynamical system represented by functions $f_1(t)\dots f_h(t)\dots f_H(t)$ satisfying a set of differential equations of the form

$$\mathbb{D}f_h = \mathbf{U}_h(f_1,\dots,f_H,\mathbb{E}f_1,\dots,\mathbb{E}f_H,), \tag{1}$$

where $\mathbb{D}$, $\mathbb{E}$ are differential operators in the temporal variables (t) and $\mathbf{U}:\mathbb{R}^J\to\mathbb{R}^H$ is an unknown map. Suppose we have a series of T equally spaced in time measurements, that is, we observe $f_h(t_i)\quad i\in\{1,\dots,T\}, h\in\{1,\dots,H\}$. An example of a dynamical system we will study in this paper is the classical Rössler system (2) [9]. This system was originally designed to have similar properties and be simpler than the Lorenz system [10] which was, at the same time, a simplified model for atmospheric convection. The system is given by

$$\begin{aligned}\frac{df_1(t)}{dt} &= -f_2(t)-f_3(t),\\ \frac{df_2(t)}{dt} &= f_1(t)+\alpha f_2(t),\\ \frac{df_3(t)}{dt} &= \beta+f_3(f_1(t)-\gamma),\end{aligned} \tag{2}$$

for constants α,β,γ. This system can be written in the form (1) by taking $\mathbf{f}=(f_1,f_2,f_3)$, $\mathbb{D}=(d/dt,d/dt,d/dt)$, and $\mathbf{U}=(U_1,U_2,U_3)$, where $U_1(v_1,v_2,v_3)=-v_2-v_3$, $U_2(v_1,v_2,v_3)=v_1+\alpha v_2$ and $U_3(v_1,v_2,v_3)=\beta+v_3(v_1-\gamma)$. For certain values of the parameters α,β,ρ, the system is known to have chaotic solutions [11,12].

Suppose now that we have access to a particular time series that was generated by this system. Our objective is to find some system of differential equations that can explain the behavior of the measurements. The SINDy method works by choosing a large dictionary of functions and regressing discretizations of $\frac{df_1}{dt},\dots,\frac{df_H}{dt}$ on the dictionary. The dictionary in question can be formed, for example, by collecting polynomial powers of f_h, $h=1,\dots,H$, spatial derivatives of $f_1\dots,f_H$, and trigonometric functions of t. A concrete simple example of such a dictionary in the case in which $H=1$ is the following:

$$\mathcal{A}=\{t,t^2,\sin(t),f_1,f_1^2,tf_1\}.$$

Of course in practice all derivatives are replaced by the corresponding finite differences taken from the measurements represented in f. Having chosen a dictionary, we let $\mathcal{A}=(A_1,\dots,A_p)$ be the vector collecting all members of the dictionary.

Using the observations of the dynamical system, a regression model can be fitted to find the combination of the elements of the dictionary of functions that adequately explains the behavior of $\frac{df_1}{dt}, \dots, \frac{df_H}{dt}$. That is, we look for a vector of regression coefficients $\mathbf{c_h} = (c_{1,h}, \dots, c_{p,h})$, such that for all $h = 1, \dots, H$,

$$\frac{df_h(t)}{dt} \approx \sum_{i=1}^{p} c_{i,h} A_i(t). \tag{3}$$

The regression model has to be learned using the available data. This regression problem could be solved in principle using least-squares. However, the ordinary least-squares regression estimator is ill-defined in cases in which the number of predictor variables p is larger than the number of observations. Since the analyst is usually uncertain about the number of elements in the dictionary needed to adequately model the system of interest, the method used to solve the regression problem at hand should allow for a large number of predictor variables (possibly larger than the number of observations) and automatically estimate sparse models, that is, generate accurate models that only use a relatively small fraction of predictor variables. The Lasso regression technique is perfectly suited for this task. The Lasso is an ℓ_1-regularized least-squares regression estimator, defined as follows. For $h = 1, \dots, H$, such that f_h is observable, we let

$$\hat{\mathbf{c}}_h = \arg\min_{\mathbf{c}_h \in \mathbb{R}^p} \sum_{l} \left(\frac{d^n f_h}{dt^n}(t_l) - \sum_{i=1}^{p} c_{i,h} A_i(t_l) \right)^2 + \lambda ||\mathbf{c}_h||_1, \tag{4}$$

where $\lambda > 0$ is a tuning constant, measuring the amount of regularization. In practice, λ is usually chosen by cross-validation. It can be shown that the ℓ_1 penalty encourages sparse solutions and that the larger $\lambda > 0$ the sparser the solution vector $\hat{\mathbf{c}}_h$ will be. In fact, there exists a large body of work studying the connection between ℓ_1 penalties and sparse estimation. For instance, assume that a linear regression model relating outcome and predictor variables holds with a sparse regression vector. More precisely, assume that the true regression vector has at most s nonzero entries, where $s \log(p) \ll n$ and n is the total sample size. Then, if λ is appropriately chosen, under certain technical conditions, the solution to the optimization problem in Eq. (4) will have approximately s nonzero entries, and its estimation error in the ℓ_2 norm will be of order $\sqrt{s \log(p)/n} \ll 1$. [See Chap. 11 of [13] (in particular Theorem 11.1) for the precise mathematical statement of these results, and the rest of the book for a full treatment of the use of Lasso and related methods for sparse estimation.] Note that any other sparse regression technique could have been used to estimate the coefficients. We prefer the Lasso due to its simplicity and the wide availability of efficient algorithms to compute it. See for example [14].

The main assumption behind this methodology is that the dynamical system that generated the data at hand can, in reality, be at least approximated using a sparse model, that is, that the vectors $\hat{\mathbf{c}}_h$ in Eq. (4) are either exactly or approximately sparse. This hypothesis is known to hold for several dynamical systems of interest in different fields of science, see [2]. If the hypothesis holds, we can expect the Lasso estimates to select only a few elements of the dictionary, namely, those that do a good job at explaining variations in the response variable [13].

It is important to notice that, even if the sparsity assumption holds, there are no guarantees that the ODE found by the algorithm will respect any motion integral of the system nor that good coefficients would be found if the analyzed ODE is near a discontinuity of the parametric map to solution. So, if extra information of the studied system is known in advance there is no way to introduce the proper constraint in the current formulation.

In [1–3] the authors propose to use an *ad hoc* linear regression estimator based on iteratively thresholding the least-squares estimator, and applied this method only to first-order systems. Through extensive numerical experiments, they show that this methodology is able to learn systems of partial differential equations that adequately model the dynamical system that generated the data. Unfortunately, if some variables are latent, that is, if one is unable to measure at least one of $f_1, \dots, f_H$, the approach described above cannot be used directly. Next, we describe a way of extending this methodology to deal with the case in which some variables are latent.

C. Our proposal: L-ODEfind

To accommodate the possibility of latent variables we note that, for a large class of dynamical systems, it is possible to reconstruct a trajectory (equivalent to the original one) given only one of the model variables, using its higher order derivatives [5,6]. Moreover, we recall that in the case of a linear system of n ordinary differential equations there is an equivalence between this multidimensional system and a single differential equation of order n, which we can interpret as latently including the information of the other $n - 1$ unobserved variables. Reference [7] also makes use of higher order time derivatives to deal with unobserved variables.

Based on these ideas, we propose to augment the methodology developed in [1–3] by choosing the target variable to be a higher-order time derivative, to tackle situations in which not all relevant variables are observed. We estimate the coefficients of the dynamical system using the Lasso estimator (4). As mentioned earlier, we chose to use the Lasso due to its simplicity, the abundance of theoretical guarantees on its performance [13] and the availability of efficient algorithms to solve the convex optimization problem that defines the estimator. The choice of the tuning constant λ in Eq. (4) is done by tenfold cross validation, using the LassoCV method from sklearn [15] with a maximum number of steps equal to 10^4, and 100 candidate λs. After estimating the regression coefficients, we build a forecasting method by integrating the retrieved differential equation. We call this method Latent ODE find (L-ODEfind).

For instance, suppose we have observed a single time series f $(H = 1)$. If we choose as a target variable the third time derivative and we use polynomial combinations up to degree 2 of the series and derivatives up to order 2 as regressors, Eq. (3) becomes

$$\frac{d^3 f(t)}{dt^3} \approx \sum_{i=1}^{p} c_i A_i(t) \tag{5}$$

and

$$\mathcal{A} = \left\{1, f, \frac{df(t)}{dt}, \frac{d^2f(t)}{dt^2}, f^2, \left(\frac{df(t)}{dt}\right)^2, \left(\frac{d^2f(t)}{dt^2}\right)^2, f\frac{df(t)}{dt}, f\frac{d^2f(t)}{dt^2}, \frac{df(t)}{dt}\frac{d^2f(t)}{dt^2}\right\}.$$

This regression model is then fitted using the Lasso (4), as described earlier.

D. Evaluation with predictions

Evaluating the equations found by these methods can be challenging. The evaluation of a method that aims at recovering the differential equation behind the observed data needs to be done in different ways depending on the information available.

In the case of a simulation where the coefficients of the differential equation are known and the variables are fully observed, the adjusted coefficients and the true ones can be compared using the mean squared error (A_t being the real value and F_t the forecasted value):

$$\mathrm{MSE}(A, F) = \frac{1}{n}\sum_{t=1}^{n}(F_t - A_t)^2.$$

If not all variables are observed, the adjusted coefficients refer to a different differential equation that in most cases cannot be obtained analytically (exceptions are, for instance, linear ODEs). In this case, the coefficients comparison cannot be done and another way of evaluating the method is needed. The same happens when working with real world data where the differential equation behind is not known. In order to evaluate a method in this context, we propose to split the observed data in two (Fig. 3). The first part is used to fit the differential equation. Then, by integrating the learned equation with initial condition taken from the end of the series used for fitting, we obtain predictions for different time horizons and compare them to the observed data (which was not used for fitting). The comparison is made by using the symmetric mean absolute percentage error (SMAPE) (regularly used in forecasting competitions such as M3 [16] and M4 [17]) as it allows a relative error comparison between predictions and observations while penalizing equally both mistakes, under and below, which is not the case for MAPE. Using a time horizon of n, the SMAPE is defined as

$$\mathrm{SMAPE}(A, F) = \frac{1}{n}\sum_{t=1}^{n}\frac{|F_t - A_t|}{(|A_t| + |F_t|)/2},$$

where A_t is the real value and F_t the forecasted value. If the predictions made by method A are better (lower values of SMAPE) than the ones made by method B, the differential equation found by the method A is deemed better than the one found by method B.

E. Implementation details

All the experiments in this paper were performed using PYTHON 3.8, except for GPoMo, which was performed using R 3.6 [18] and the GPoM package [19]. For L-ODEfind we use our own PYTHON implementation (available at [20]). The integration of the differential equations was done using ODEINT from the PYTHON Scipy library [21].

III. RESULTS

In this section we first compare L-ODEfind with GPoMo for the problem of learning an ordinary differential equation with no hidden variables. Then, we compare the performance of these methods in simulated systems with hidden variables and finally in temperatures series provided by RTE.

A. Names abbreviation

The names abbreviation used in this section can be found in Table I. Target time derivative is the degree of the fitted differential equation and poly degree is the maximum degree of the polynomial combinations of the derivatives used as regressors. An example of the regression problem with target derivative 3 and poly degree 2 can be found in Eq. (5). Notice that the number in the model name refers to the degree of the differential equation.

B. Simulated data with fully observed variables

We compare the performance of L-ODEfind with that of GPoMo for the task of learning an ordinary differential equation with no hidden variables. Note that since in this case there are no hidden variables, our method coincides with SINDy. The goal of this comparison is to highlight the fact that, because L-ODEfind solves a continuous optimization problem and GPoMo approximately solves a combinatorial optimization problem, L-ODEfind can be orders of magnitude faster that GPoMo.

We generated data using the Lorenz attractor equations:

$$\begin{aligned}\frac{dx}{dt} &= \sigma(y - x),\\ \frac{dy}{dt} &= x(\rho - z) - y,\\ \frac{dz}{dt} &= xy - \beta z,\end{aligned} \tag{6}$$

TABLE I. Models names abbreviations used in the graphics above.

Method	Parameters	Name
L-ODEfind	Target derivative: 1, poly degree: 3	L-odefind1
L-ODEfind	Target derivative: 2, poly degree: 3	L-odefind2
L-ODEfind	Target derivative: 3, poly degree: 3	L-odefind3
GPoMo	Target derivative: 2, poly degree: 3	GPoMo2
GPoMo	Target derivative: 3, poly degree: 3	GPoMo3

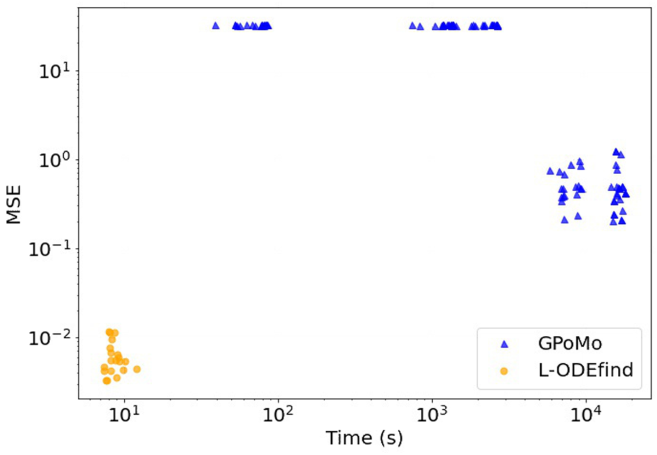


FIG. 1. Comparison between L-ODEfind and GPoMo when finding the coefficients of data generated by the Lorenz attractor differential equations. We plot the average MSE across different random initial conditions for the system, and the time (in seconds) needed by the algorithm to fit the model.

using the coefficients $\sigma = 10$, $\rho = 28$, and $\beta = \frac{8}{3}$, for 20 different random starting conditions, with distribution $N(0, 1)$. The differential equation was integrated using a discretization step of 0.01. Then, we fitted the resulting datasets using GPoMo and L-ODEfind. For GPoMo we set a maximum number of steps of 10240. We compare the true coefficients of the differential equations with the ones found by each model using as a metric the mean square error (MSE). All the coefficients were taken into account for computing the MSE, including the ones that are supposed to be zero. We average the MSE corresponding to different random starting conditions and report this as our goodness of fit metric.

Figure 1 shows that L-ODEfind method is nearly two orders of magnitude more accurate than the best GPoMo case. On the other hand, L-ODEfind is also orders of magnitude faster, taking less than 10 s to compute accurate approximations of the true coefficients while GPoMo takes hours.

C. Simulated data with hidden variables

In this section we use three ODE systems, an oscillator, and the Rössler and Lorenz systems, as examples to evaluate and compare, using the methodology explained in Sec. II D, the accuracy of several L-ODEfind and GPoMo models. For a given differential equation, we generated time series of length 5000 points and integration step of $dt = 0.01$ for 20 different initial conditions. Following the methodology described in Sec. II D we fit each series with GPoMo or L-ODEfind and then integrate the equation found in each case to predict the values of the series for several time horizons and compute the corresponding SMAPE. The number of maximum iterations for GPoMo is set to 5120.

1. Example 1: Oscillator

We start by considering an oscillator, which is a first order linear system of two variables. An equation of order two involving only one of the variables can be derived. So, in this case, the problem of having hidden variables can be effectively tackled by choosing a higher order time derivative (order two) as the target.

We used the two variables oscillator equation which can be written in general form as

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}, \tag{7}$$

where x and y are the variables and a, b, c, d are the coefficients of the linear equation that links the variable with its derivatives. If we only had access to the variable x, this system could be rewritten in a second order differential equation taking the form

$$\frac{d^2x}{dt^2} = (a + d)\frac{dx}{dt} - (ad - bc)x = \beta\frac{dx}{dt} + \alpha x. \tag{8}$$

In our experiments we set $a = 0.1, b = -1, c = 1, d = 0$ ($\alpha = -1, \beta = 0.1$) and we only observed the variable x so the corresponding second order equation derived as in Eq. (8) is $0.1\frac{dx}{dt} - x$. We can see in Fig. 2(a) that using the second derivative in time as a target gives the lowest prediction error

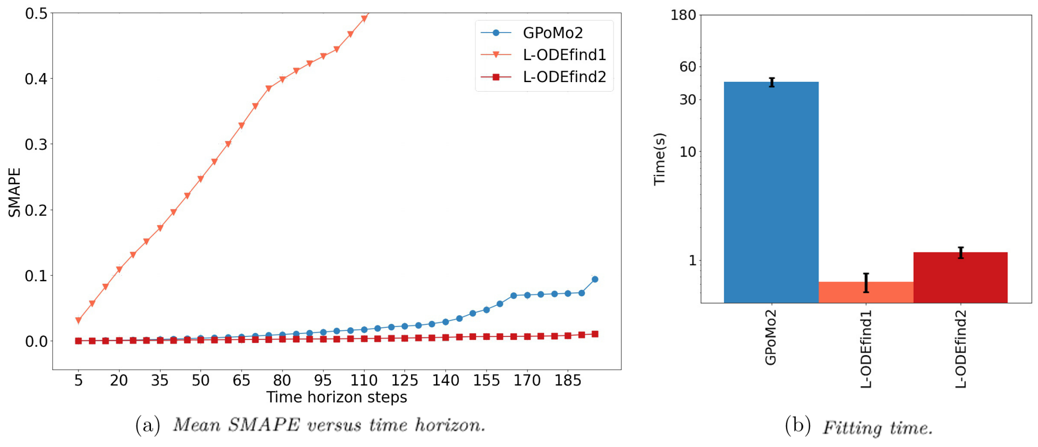
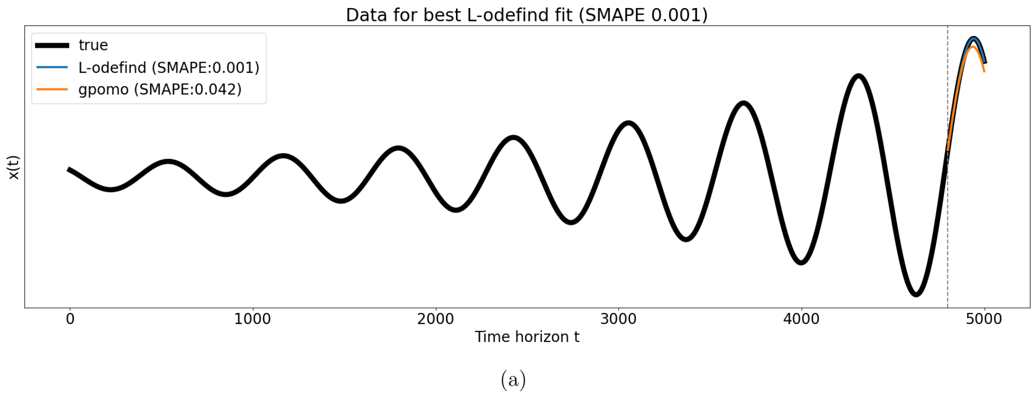
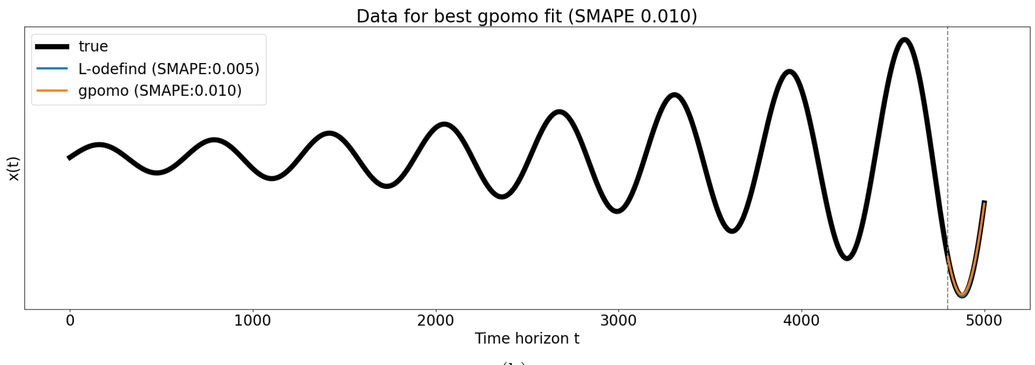


FIG. 2. Prediction accuracy and fitting time for L-ODEfind and GPoMo models when data come from the oscillator system with x as observed variable (y hidden).



(a)

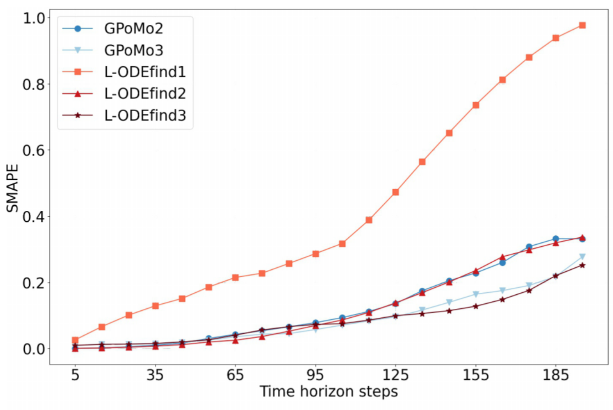


(b)

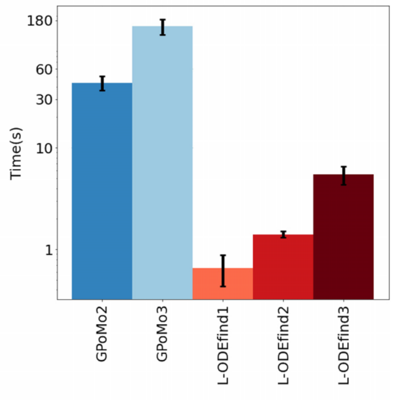
FIG. 3. Predictions for the oscillator system with x as observed variable. In black are the true data. To the left of the dotted line is the data that was used for fitting the models (inferring the coefficients of the ODE). In orange are the GPoMo predictions and in blue the L-ODEfind predictions.

for both GPoMo and L-ODEfind, as expected from Eq. (8). We also see that the model found by L-ODEfind manages to approach the true model much better than GPoMo as it maintains an SMAPE below 0.02 in horizons were GPoMo has already arrived to 0.1. As an example to perceptually evaluate the meaning of this lower SMAPE, in Fig. 3 the true data and the GPoMo and L-odefind fits are shown. For both models we see that the predictions are almost indistinguishable from the true solution with GPoMo sometimes falling slightly apart.

When looking at the coefficients found by GPoMo ($\alpha = -0.995 \pm 0.004$ and $\beta = 0 \pm 0$) and L-ODEfind ($\alpha = -0.9997 \pm 0.0003$ and $\beta = 0.0996 \pm 0.0001$) are within a small tolerance the expected from Eq. (8). Finally, when looking at the fitting time [Fig. 2(b)], we find that L-ODEfind is around 50 times faster than GPoMo.



(a) *Mean SMAPE versus time horizon.*



(b) *Fitting time.*

FIG. 4. Prediction accuracy and fitting time for L-ODEfind and GPoMo models when data comes from the Rössler system with y as observed variable (x and z hidden).

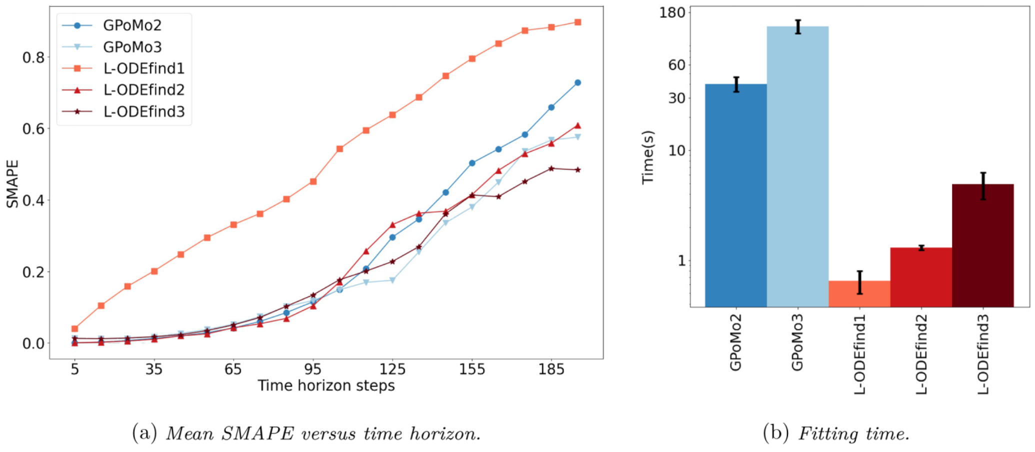


FIG. 5. Prediction accuracy and fitting time for L-ODEfind and GPoMo models when data comes from the Rössler system with x as observed variable (y and z hidden).

2. Example 2: Rössler

Next, we consider a more complicated case, the Rössler system, which is a nonlinear (quadratic) system:

$$\frac{dx}{dt} = -x - z,$$
$$\frac{dy}{dt} = x + ay,$$
$$\frac{dz}{dt} = b + z(x - c). \tag{9}$$

If only variable y is observed (x and z hidden) an equation of order 3 and polynomial degree 2 can be deduced for y [22].

In our experiments, we set $a = 0.52, b = 2, c = 4$, and only observe y (x and z latent). L-ODEfind is consistently faster than GPoMo [Fig. 4(b)] although, in this example, GPoMo3 and L-ODEfind3 have almost the same prediction SMAPE [Fig. 4(a)]. Notice that the SMAPE is less than 0.2 for all models with target time derivative 2 or 3 up to 125 time horizon steps.

Therefore, when considering systems where an analytical solution can be deduced, such as the oscillator (x observed, y hidden) and Rössler system (y observed, x and z hidden), both methods perform very well in terms of SMAPE prediction.

Next, we consider the Rössler system in the case of variable x observed (y and z hidden). In this scenario, there is no analytical solution using only polynomials [22]. Interestingly, we can see that using higher order time derivatives as target, both GPoMo and L-ODEfind find a differential equation that is able to provide predictions whose accuracy is comparable (for short-term horizons) to the previous case (y observed) where there was an analytical solution (Fig. 5). The fitting times continue to show that L-ODEfind is consistently faster than GPoMo.

3. Example 3: Lorenz attractor

Here, we consider the Lorenz system [10], discussed in Sec. III B, where no differential equation using only polynomials can be derived for x as the only observed variable.

Following the same methodology as before, we tried different target derivatives for both GPoMo and L-ODEfind to fit observed variable x (y and z served). We found that using higher order time derivatives (in particular second order) helps to find models that can approximate better the observed

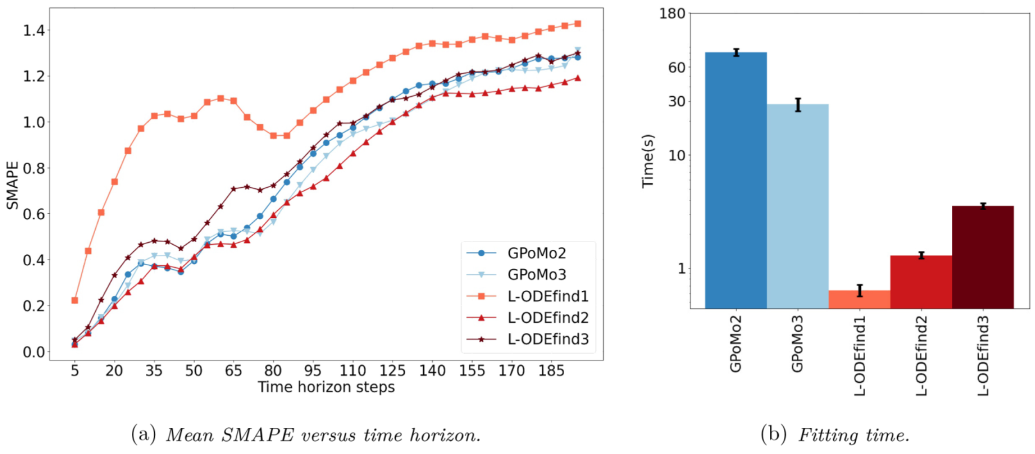
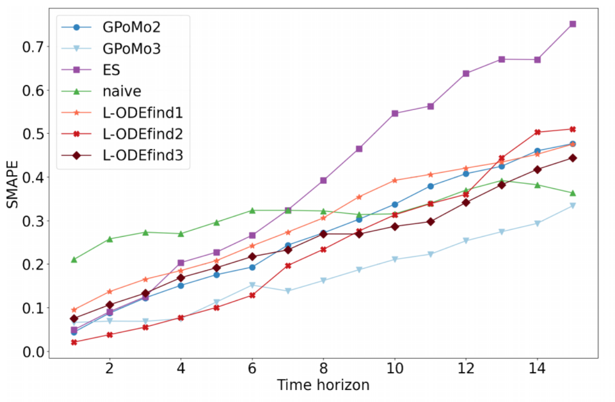
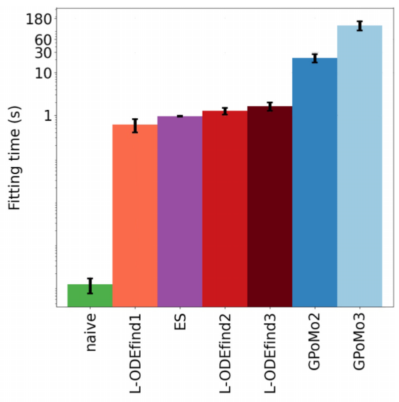


FIG. 6. Prediction accuracy and fitting time for L-ODEfind and GPoMo models when data comes from the Lorenz system with x as observed variable (y and z hidden).



(a) *SMAPE vs time horizons in hours for different predictive methods.*



(b) *Time to fit the different models in log scale.*

FIG. 7. SMAPE and fitting times for different models applied on a real world data set of temperatures.

time series when integrated, although the prediction accuracy suffers from the added complexity of the problem (Fig. 6). As a consequence, the accuracy degrades faster reaching SMAPE = 0.5 as soon as 35 time steps while in the previous cases this was attained around 185 steps. For a narrow difference again, L-ODEfind outperformed GPoMo while also keeping fitting times 4 to 40 times faster than GPoMo.

D. Temperature series provided by RTE

In this section, we fit different models to temperature series provided by the Réseau de Transport d'Électricité (RTE). The data consist of 200 hourly measured temperatures time series. These time series correspond to temperatures in Paris along a year for 200 different possible years or scenarios. The time series are not measured temperatures nor the output of a simulation, but rather the result of a reanalysis process. Also, some relevant variables for modeling the atmospheric system are not available to us, for instance, wind and pressure. Our temperature time series are not historical measurements but can be thought as a possible realization of the temperature in Paris. These temperature time series have $365 \times 24 = 8760$ time points.

In order to evaluate the methods, we select 39 out of the 200 temperature time series (due to GPoMo's computation time), fit the different models and use them to predict for short time horizons. The fit was done with the first 8560 time points and the prediction was evaluated with the following time points (up to a time horizon of 15 time points).

In tackling this complex problem we want to analyze the performance of L-ODEfind and GPoMo in the forecasting task and compare their behavior to classical forecasting methods naive predictor (for every time horizon, predicts the average of the last 24 time points) and exponential smoothing (ES) (triple exponential smoothing with an additive seasonal component[1]).

[1]We used the implementation available in the PYTHON library sktime [23].

In Fig. 7 the SMAPE of the prediction for different time horizons in hours is displayed. L-ODEfind2 gives the lowest SMAPE for time horizons lower or equal to 6 h, whereas GPoMo3 is better for time horizons greater than 6 h. Notice that for time horizons greater than 6 h, the naive predictor performs better than exponential smoothing, giving a rough idea of the reasonable predictability horizon that forecasting methods can give. In any case, both L-ODEfind and GPoMo give better forecasts than both ES and naive when the target derivative is 3. As can be seen in Fig. 7(b), the fitting times for GPoMo are much greater than L-ODEfind, whereas L-ODEfind and exponential smoothing have similar fitting times.

IV. DISCUSSION

In this paper we addressed the problem of recovering differential equations from data where not all variables are observed by enhancing the approach outlined in [1]. After testing it in simple and complex ODE systems, we consistently found that the proposed method of using time derivatives of higher order as target regressing variables allows to find models whose future predictions are more reliable than only using first order derivatives. We also compared our method to GPoMo and found that: (i) our proposal is orders of magnitude faster that GPoMo, and (ii) our proposal learns models with comparable or even higher prediction accuracy for several dynamical systems. Finally, we faced the challenge of addressing a real world problem and found that both L-ODEfind and GPoMo used as forecasting methods gave comparable results while at the same time outperforming classical forecasting methods, exponential smoothing and naive. In summary, L-ODEfind proved to be an accurate and fast method for recovering ordinary differential equations from data with hidden variables.

ACKNOWLEDGMENTS

This work was supported by Aristas S. R. L. and RTE (Réseau de Transport d'Électricité).

[1] S. L. Brunton, J. L. Proctor, and J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. USA **113**, 3932 (2016).

[2] M. Quade, M. Abel, J. N. Kutz, and S. L. Brunton, Sparse identification of nonlinear dynamics for rapid model recovery, Chaos **28**, 063116 (2018).

[3] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, Data-driven discovery of partial differential equations, Sci. Adv. **3**, 26 (2017).

[4] S. Kida, M. Yamada, and K. Ohkitani, in *Route to Chaos in a Navier-Stokes Flow*, edited by M. Mimura and T. Nishida, *Recent Topics in Nonlinear PDE IV*, Vol. 160 of North-Holland Mathematics Studies (North-Holland, Amsterdam, 1989), pp. 31–47.

[5] R. Mañé, On the Dimension of the Compact Invariant Sets of Certain Non-linear Maps, *Dynamical Systems and Turbulence* (Springer-Verlag, New York, 1981), p. 230.

[6] F. Takens, Detecting strange attractors in turbulence, in *Dynamical Systems and Turbulence, Warwick 1980*, edited by D. Rand and L. S. Young, Lecture Notes in Mathematics (Springer, Berlin, Heidelberg, 1981), Vol. 898, pp. 366–381.

[7] S. Mangiarotti, R. Coudret, L. Drapeau, and L. Jarlan, Polynomial search and global modeling: Two algorithms for modeling chaos, HAL **86** (2012).

[8] J. Bongard and H. Lipson, Automated reverse engineering of nonlinear dynamical systems, Proc. Natl. Acad. Sci. USA **104**, 9943 (2007).

[9] O. Rössler, Chaotic behavior in simple reaction systems, Z. Naturforsch. A **31**, 259 (1976).

[10] E. N. Lorenz, Deterministic nonperiodic flow, J. Atmos. Sci. **20**, 130 (1963).

[11] G. Amaral, C. Letellier, and L. Aguirre, Piecewise affine models of chaotic attractors: The Rössler and Lorenz systems, Chaos **16**, 013115 (2006).

[12] K. M. Ibrahim, R. K. Jamal, and F. H. Ali, Chaotic behaviour of the Rössler model and its analysis by using bifurcations of limit cycles and chaotic attractors, J. Phys.: Conf. Ser. **1003**, 012099 (2018).

[13] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning with Sparsity: The Lasso and Generalizations* (Chapman and Hall/CRC, Boca Raton, FL, 2015).

[14] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani, Pathwise coordinate optimization, Ann. Appl. Stat. **1**, 302 (2007).

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. **12**, 2825 (2011).

[16] S. Makridakis and M. Hibon, The M3-competition: Results, conclusions and implications, Int. J. Forecast. **16**, 451 (2000).

[17] J. L. Castle, J. A. Doornik, and D. Hendry, Some forecasting principles from the M4 competition, Economics Papers 2019-W01, Economics Group, Nuffield College, University of Oxford, Jan. 2019, https://ideas.repec.org/p/nuf/econwp/1901.html.

[18] R. Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2020, https://www.R-project.org/.

[19] S. Mangiarotti, M. Huc, F. L. Jean, M. Chassan, and L. Drapeau, GPoM: Generalized Polynomial Modelling, 2020, R package version 1.3 (developed by CESBIO), https://mran.microsoft.com/snapshot/2022-03-23/web/packages/GPoM/index.html.

[20] https://github.com/agussomacal/L-ODEfind.

[21] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, and P. van Mulbregt, SciPy 1.0 contributors, SciPy 1.0: Fundamental algorithms for scientific computing in Python, Nat. Methods **17**, 261 (2020).

[22] C. Letellier, L. A. Aguirre, and J. Maquet, Relation between observability and differential embeddings for nonlinear dynamics, Phys. Rev. E **71**, 066213 (2005).

[23] M. Löning, A. Bagnall, S. Ganesh, V. Kazakov, J. Lines, and F. J. Király, sktime: A Unified Interface for Machine Learning with Time Series, in *33rd Conference on Neural Information Processing Systems* (NeurIPS, Vancouver, Canada, 2019).