Data Driven Discovery of Cyber Physical Systems

Ye Yuan[1,2], Xiuchuan Tang[3], Wei Zhou[1], Wei Pan[4], Xiuting Li[1],

Hai-Tao Zhang[1,2], Han Ding[2,3,*] and Jorge Goncalves[1,5,6,*]

[1]School of Artificial Intelligence and Automation,

Key Laboratory of Image Processing and Intelligent Control,

Huazhong University of Science and Technology, Wuhan 430074, People's Republic of China

[2]State Key Lab of Digital Manufacturing Equipment and Technology,

Huazhong University of Science and Technology, Wuhan 430074, People's Republic of China

[3]School of Mechanical Science and Engineering,

Huazhong University of Science and Technology, Wuhan 430074, People's Republic of China

[4]Department of Cognitive Robotics, Delft University of Technolog, Delft, Netherlands

[5] Department of Engineering, University of Cambridge, CB2 1PZ, United Kingdom

[6]Luxembourg Centre for Systems Biomedicine, University of Luxembourg,

7 Avenue des Hauts Fourneaux, 4362, Esch-sur-Alzette, Luxembourg

*Corresponding Authors. E-mail: dinghan@hust.edu.cn, jmg77@cam.ac.uk.

Abstract

Cyber-physical systems embed software into the physical world. They appear in a wide range of applications such as smart grids, robotics, intelligent manufacture and medical monitoring. Cyber-physical systems have proved resistant to modeling due to their intrinsic complexity arising from the combination of physical components and cyber and the interaction between them. This study proposes a general framework for reverse engineering cyber-physical systems directly from data. The method involves the identification of physical systems as well as the inference of transition logics. It has been applied successfully to a number of real-world examples ranging from mechanical and electrical systems to medical applications. The novel framework seeks to enable researchers to understand the underlying mechanism of cyber-physical systems as well as make predictions concerning the trajectory of cyber-physical systems based on the discovered model. Such information has been proven essential for the assessment of the performance of cyber-physical systems; it can potentially help debug in the implementation procedure and guide the redesign to achieve the required performance.

INTRODUCTION

Since the invention of computers, software has quickly become ubiquitous in our daily lives. Software controls domestic machines, such as washing and cooking appliances, aerial vehicles such as quadrotors, the scheduling of power generation and the monitoring of human body vital signals. These technologies embed cyber components throughout our physical world. In fact, almost all modern engineering systems involve the integration of cyber and physical processes. The integration of cyber and physical components provides new opportunities and challenges. On one hand, this integration produces new functionality in traditional physical systems, such as brakes and engines in vehicles, intelligent control systems for biochemical processes and wearable devices [1–3]. On the other hand, the integration of cyber components adds new layers of complexity, potentially seriously complicating their design and guaranteeing their performance. Cyber-Physical Systems (CPSs), such as modern power grids or autonomous cars, require guarantees on performance to be economically and safely integrated into society. In power grids, the failure of transformer taps, capacitors and switching operations alter the dynamics of the grid, which can be extremely costly. We have, after all, already witnessed a massive power outage in Southern California on September 2011 due to a cascading failure from a single line tripping (which was not detected by operators using their model), costing billions of US dollars. In autonomous driving, autonomous cars are expected to be well-operated in multiple complex scenarios from driving on multi-lane highway to turning at intersections while obeying rules. These objectives are achieved through decisions made by high-level software and control by low-level computer systems, realizing the command using a combination of GPS/IMU, camera, radar and LIDAR data [4]. In such complex scenarios, guaranteeing CPS's performance poses a fundamental challenge.

For performance guarantees, we require reliable models that capture essential dynamics. The central question this study seeks to answer, therefore, is how to reliably and efficiently automate mechanistic modeling of CPSs from data [6, 7]. An appropriate mathematical model of CPS should recognize the hybridity of CPS, which comprise of discrete and continuous components due to the integration of software and physical systems, respectively. Hybrid dynamical systems (detailed below and Supplementary Section) use finite-state machines to model the cyber components and dynamical systems for the physical counterparts. Hybrid dynamical models can produce accurate predictions and enable assessments of the

CPSs' performance [8]. This paper presents a new method, namely Identification of HYbrid Dynamical Systems (IHYDE), for automating the mechanistic modeling of hybrid dynamical systems from observed data. IHYDE has low computational complexity and is robust to noise, enabling its application to real-world CPS problems.

There are various methods for identifying non-hybrid dynamical systems. Schmidt and Lipson [9] proposed a data-driven approach to determine the underlying structure and parameters of time-invariant nonlinear dynamical systems. Schmidt and Lipson's method uses symbolic regression to identify the system, balancing model complexity and accuracy. However, symbolic regression has its possible limitations: it can be computationally expensive, does not scale to real large-scale systems, and is prone to overfitting. The work in [10–12] expanded the vector field or map of the underlying system into suitable function series; then, they used compressive sensing and sparse Bayesian learning techniques to accurately estimate various terms in the expansion. Later, Brunton et. al. [22] applied a sequentially thresholded least-square method to discover ordinary differential equations. Although these recent advances [10–12, 22] managed to reduce the expensive computational burden using compressive sensing and sparse learning, these methods cannot be applied to hybrid dynamical systems because of the complexity and switching behaviors in hybrid dynamical systems; basically, these algorithms cannot account for an unknown number of unknown subsystems that interact via unknown transition logic.

There have been a number of interesting results in hybrid dynamical system identification over the past two decades [14–16, 18–25]. Table I summarizes key characteristics of a subset of important methods. Reference [15] gives a comprehensive literature review, summarizing major advances up to 2007. Methods span across several fields, such as algebraic geometry [16], mixed integer programming [19], bounded-error [20], Bayesian learning [21], clustering-based strategies [22], and multi-modal symbolic regression [23]. The algebraic geometry [16] and bounded-error [20] methods can handle cases with unknown model order and number of subsystem models. However, algebraic-geometric methods cannot deal with discontinuous dynamics. The Bayesian approach in [21] exploits available prior knowledge about modes and parameters of hybrid systems, which helps tuning parameters. Clustering-based methods [22] are suitable for cases with little prior knowledge on physical systems. However, it requires prior knowledge of model order and number of subsystems. Despite clear merits of all these pioneering contributions, most methods focus on the simplest hybrid

3

dynamical models: piecewise affine systems with linear transition rules [20] (see Table I).

Recent pioneering results in [18, 23] use compressive sensing [26] to identify the minimum number of subsystems by recovering a sparse vector-valued sequence from data. These algorithms tradeoff the mismatch between data and model predictions, and the energy of the noise. Breschi et. al. [37] proposes a regression approach based on recursive clustering and multi-class linear separation methods. To solve time-varying parameter identification on stochastic autoregressive models with exogenous inputs (ARX), [38] employs expectation maximization algorithms to recursively solve mixed-integer optimizations problems. The work in [39] proposes a method based on difference of convex functions programming to optimize non-smooth and nonconvex objective functions. Finally, [40] aims to identify switched affine models in a set membership framework, [41] uses hybrid stable spline algorithms, where Gaussian processes model the impulse response of each subsystem, and [24] uses symbolic regression.

IHYDE aims to provide a more flexible and general framework by directly discovering the number of subsystems, their dynamics, and the transition logic from data (see Table I). IHYDE deals with this problem in two parts: first, the algorithm discovers how many subsystems interact with each other and identifies a model for each one; second, the algorithm infers the transition logic between each pair of subsystems. Methods in [10, 22, 23] can be viewed as special cases of the first step in this new framework. IHYDE is illustrated on a number of examples, ranging from power engineering and autonomous driving to medical applications, to demonstrate the algorithm's application to various types of datasets.

RESULTS

The IHYDE algorithm

This section is divided in two major parts. The first presents the proposed inference-based IHYDE algorithm using a simple example – a thermostat, while the second illustrates its applicability to a wide range of systems, from real physical systems to challenging in silico systems, and from linear to nonlinear dynamics and transition rules. Details of both the algorithm and how data was acquired or generated can be found in Supplementary Materials.

TABLE I. A summary of key characteristics of major state-of-the-art methods. The first column lists different methods in the literature, while the remaining three columns list their characteristics. Specifically, the second column shows the required prior knowledge on the number of subsystems. The last two columns specify the model classes of the subsystem dynamics and transition logics that these methods can be applied to, respectively.

Methods	Prior knowledge on # of subsystems	Classes of dynamics	Classes of transition logics
[16]	unknown	switched ARX systems	none
[23]	unknown	switched ARX systems	none
[22]	given	piecewise affine ARX systems	hyperplane
[21]	given	piecewise affine ARX systems	hyperplane
[20]	unknown	piecewise affine ARX systems	hyperplane
[37]	unknown	piecewise affine ARX systems	polyhedral
[38]	unknown	switched ARX systems	none
[40]	unknown	switched ARX systems	none
[39]	given	switched ARX systems	none
[24]	given	hybrid nonlinear systems	nonlinear
IHYDE	unknown	hybrid nonlinear systems	nonlinear

The inference-based IHYDE algorithm applied to a thermostat. This section explains the key concepts of IHYDE using one of the simplest and ubiquitous hybrid dynamical systems: a room temperature control system consisting of a heater and a thermostat. The objective of the thermostat is to keep the room temperature $y(t)$ near a user specified temperature. At any given time, the thermostat can turn the heater on or off. When the heater is off, the temperature dissipates to the exterior at a rate of $-ay(t)$ degrees Celsius per hour, where $a > 0$ is related to the insulation of the room. When the heater is on, it provides a temperature increase rate of $30a$ degrees Celsius per hour (Fig. 1a).

Assume a desired temperature is set to 20 degrees Celsius. Thermostats are equipped with
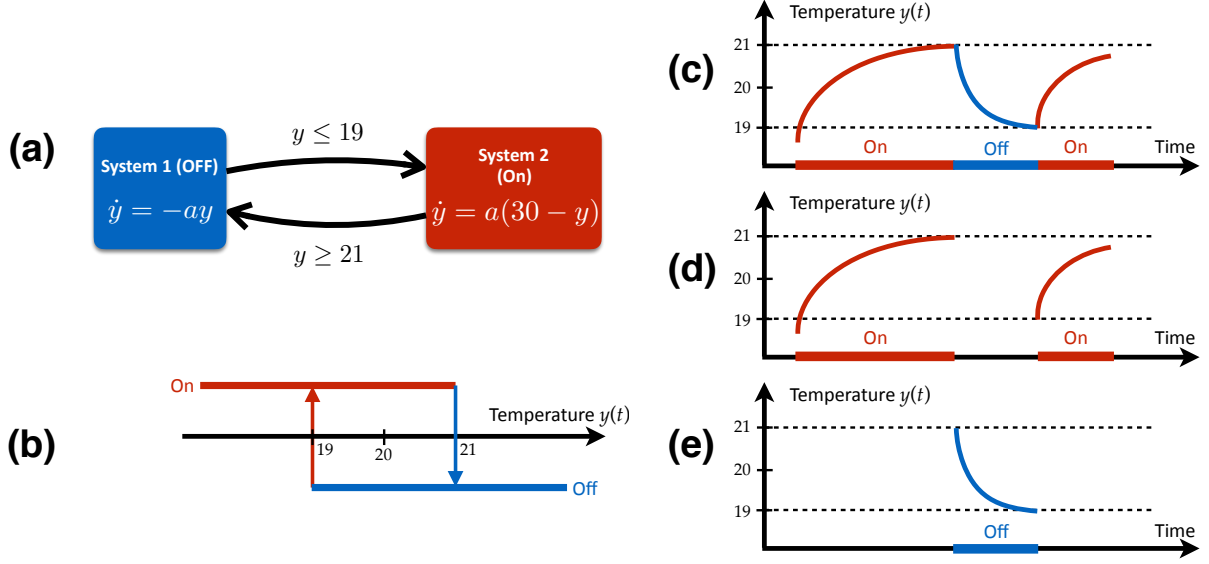
FIG. 1. An illustrative toy example on a thermostat. **(a)** The physical dynamic equations plus the transition rules of the hybrid dynamical system. A transition rule is to turn the heater on when the temperature falls below 19 degrees, and switch it off when it reaches 21 degrees. When the heater is off, the temperature y dissipates to the exterior at a rate of $-ay(t)$ degrees Celsius per hour, where $a > 0$ is related to the insulation of the room. When the heater is on, it provides a temperature increase rate of $30a$ degrees Celsius per hour. **(b)** Visualization of transition rules of the relay hysteresis based on the temperature of the room. **(c)** A simulation of the temperature of the thermostat system. Red (blue) is associated with the heater on (off). **(d) (e)** Separated time series of the temperature corresponding to the heater on (off) from the original temperature data.

hysteresis to avoid chattering, i.e., fast switching between on and off. A possible transition rule is to turn the heater on when the temperature falls bellow 19 degrees, and switching it off when it reaches 21 degrees (Fig. 1b). The goal of IHYDE is to infer both subsystems plus the transition logic from only the observed time-series data of the temperature (Fig. 1c).

Inferring subsystems. The first step of the algorithm is to iteratively discover which subsystem of the thermostat generated which time-series data. Initially, the algorithm searches for the subsystem that captures the most data, since this subsystem would explain the largest amount of data. In this case, the algorithm would firstly discover subsystem 2 (heater on) since more than half of the data corresponds to that subsystem (see Fig. 1c).

6

The time-series portion of the data (Fig. 1d) is then used to find the dynamics of subsystem 2. The algorithm is then repeated on the remaining data (Fig. 1e). In this case, there is only one subsystem left (heater off). Hence, the algorithm classifies all the rest data to a subsystem and identifies the corresponding dynamics.

Inferring transition logic. The second and final step is to identify the transition logics between the two subsystems, i.e., what triggered the transitions from on to off and from off to on. Starting with subsystem 2 (heater on) and its associated data in Fig. 1d, the algorithm first learns that no switch occurs when the temperature changes from just below 19 to near 21. Since the switch happens when the temperature reaches 21 degrees, the algorithm concludes that the switch from on to off happens when $y(t) = 21$ degrees. In practice, however, the software detects the switches when $y(t) \geq 21$. Similarly, from Fig. 1e, the algorithm learns that the switch from on to off happens when $y(t) \leq 19$.

In summary, IHYDE automatically learns the dynamics of all subsystems and the transition rules from one subsystem to another. While this is a simple system, as we will show next, this is true even in the presence of a large number of subsystems, potentially with nonlinear dynamics and transition rules.

Universal applications

Next, we illustrate how IHYDE can be applied to a wide range of applications, from power engineering to robotics to medicine, showing the flexibility, applicability and power of IHYDE to model complex systems. Here, we consider the following systems. 0) Benchmark examples (see Supplementary Example 1, Example 2, Example 3 and Example 4); 1) Autonomous vehicles and robots: design and validation of an autonomous vehicle (see Supplementary Example 5); 2) Complex electronics: Chua'a circuit (see Supplementary Example 6); 3) Monitoring of industrial processes: monitoring a wind turbine (see Supplementary Example 7); 4) Power systems: transmission lines and smart grids (see Supplementary Example 8 and Example 9); 5) Medical applications: heart atrial AP monitoring (see Supplementary Example 10). To test IHYDE's performance, these systems will include both real experiments and synthetic datasets. Details can be found below and in Supplementary Examples.

Table II contains a summary of the most important systems analyzed in the paper. The

first three examples are based on real experimental data, while the other three are based on simulated data. The first two columns illustrates the systems and the corresponding subsystems respectively where each subsystem is associated with a particular color. The third column shows the original time-series data (dots) in the color associated with the subsystem that generated it, the fitted data from the identified models (lines connecting the dots), and the location of the transitions (changes in colors). Note that, at this resolution, the original data and the data obtained from the fitted models are indistinguishable. The last column presents the relative error ratio [21] between the true data and the data simulated by the fitted model. A small error ratio indicates a good agreement between the true and modeled systems, and serves as a measure of the performance of IHYDE. Data is either collected (real systems) or simulated (synthetic systems) and captures all key transitions. As seen in column 3 and column 4 of Table II, IHYDE successfully modeled the original dynamics that generated the data in all examples with extremely high precision (nearly zero identification errors). First, it was able to classify each time point according to the respective subsystem that generated it. Second, it identified the dynamics of each subsystem with a very small error (less than 0.3% on all simulated examples). Finally, it correctly identified the transition rules between subsystems.

Autonomous vehicles and robots: design and validation. To demonstrate IHYDE's usefulness in designing and validating complex systems, we tested the algorithm on an autonomous vehicle, custom built in the lab (Table IIA). Typically, the design process of complex systems consists of an arduous, time-consuming, and trial-and-error based approach: start from an initial design, evaluate its performance and revise it until the performance is satisfied. A primary issue with this iterative approach is that when a design fails to meet desired specifications, many times engineers have little to no insight on how to improve the next iteration. Often, an engineer cannot discern whether the issue is due to poor mechanical design, issues with the software, or factors that were not considered. And this is also true with other general complex systems that involve interactions between physical/mechanical parts and software.

The autonomous electrical car consists of a body, a MK60t board, a servo motor, a driving motor, and a camera. The design goal of the autonomous car was to successfully run through a winding track as quickly as possible. Using an embedded camera, the software captures information of the upcoming road layout to ascertain whether a straightway or a
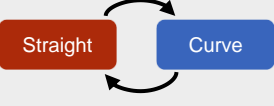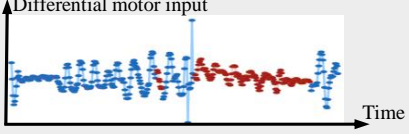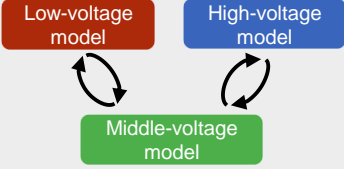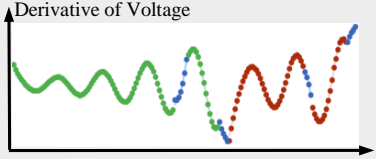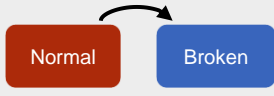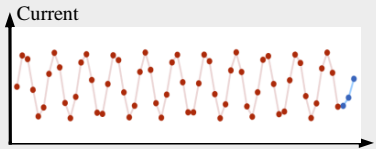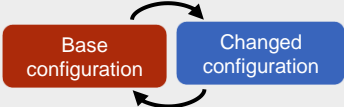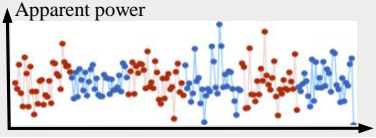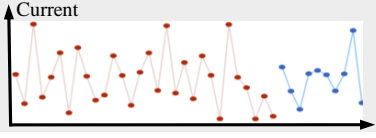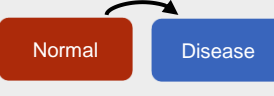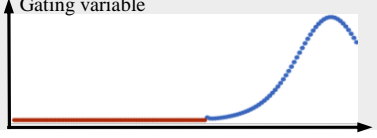
System	Hybrid dynamical System	Data fitting and transitions	Relative error ratio (%)
A Autonomous vehicles and robots	Straight ⟷ Curve	Differential motor input vs Time	0.24%
B Large scale electronics	Low-voltage model, High-voltage model, Middle-voltage model	Derivative of Voltage	5.2%
C Monitoring of industrial processes	Normal → Broken	Current	2.5%
D Smart grid	Base configuration ⟷ Changed configuration	Apparent power	0.000081%
E Power systems fault monitoring	Normal → Line Fault	Current	0.00080%
F Medical applications	Normal → Disease	Gating variable	0.029%

TABLE II. Summary of IHYDE algorithm applied to numerous examples. IHYDE has been applied to six examples in different applications. The first column illustrates the systems, while the second column shows the different subsystems plus the transition rules. Each subsystem is associated with a particular color. The third column shows the original time-series data (dots) in the color associated with the subsystem that generated it, the fitted data from the identified models (lines connecting the dots), and the location of the transitions (changes in colors). The last column presents the relative error ratio [21] between the true data and the data simulated by the fitted model. A small error ratio indicates a good agreement between the true and discovered systems, and serves as a measure of the performance of IHYDE.

curve is coming up. Based on this information, the motor chooses an appropriate power to match the desired speed control strategy. For the purpose of illustration, we considered a simple controller that provides higher velocities on straightways and lower velocities on

curves. In addition, simple feedback controllers help the car follow the chosen speed and stay on the track. The speed control strategy is based on incremental proportional and integral (PI) control that keeps the car at the correct speed, while the switching rule decides on the correct speed depending on whether a straight or curve is coming up.

For the first-round design, we deliberately swapped the straightway and curve speeds to mimic a software bug. As a consequence, the car travelled rather slow in the straights and left the tracks in the curves (Supplementary movie S1). While in this case it was rather easy for engineers to spot the software bug; debugging, in general, can be extremely difficult, sometimes only possible by trial and error, and, as a consequence, very time consuming. One would like to check whether these types of bugs could be detected by IHYDE. Indeed, from the data generated by the faulty system, IHYDE compared the discovery models with the to-be-built ones, pinpointed the incorrect speed controllers. Hence, from data alone, IHYDE successfully reverse engineered the control strategy of the CPS and discovered software bug.

Large scale electronics: Chua's circuit. Debugging and verifying large scale electronics can be a daunting experience. Modeling could help identify whether a device has been built according to the desired specifications by identifying faulty connections or incorrect implementations. Simple electrical circuits, such as RLC circuits, are linear and easy to model. However, most electronic circuits introduce both nonlinear dynamics and switches (e.g., diodes and transistors), which can lead to extremely complex behaviors. Thus, modeling such systems can be very challenging.

To illustrate IHYDE's applicability in this scenario, we built an electronic circuit that exhibits complex behaviors. We chose a well known system, called the Chua's circuit [28], that exhibits chaotic trajectories (Table IIB). Chaotic systems constitute a class of systems that depend highly on initial conditions, and makes simulation and modeling very challenging. Our circuit consists of an inductor, two capacitors, a passive resistor and an active nonlinear resistor, which fits the condition for chaos with the least components. The most important active nonlinear resistor is a conceptual component that can be built with operational amplifiers and linear resistors. The resulting nonlinear resistor is piecewise linear, making the Chua's circuit a hybrid dynamical system with a total of three subsystems and a well-defined transition logic.

After collecting real data measured from the circuit, IHYDE successfully captures the dynamics of system and the transition rules between identified subsystems. In particular,

the nonlinear dynamics are consistent with the true parameters of the circuit elements. As with all examples, modeling of the Chuas circuit was achieved using only the data and the basic knowledge of the field (to guide the choice of nonlinearities), without other assumptions on dynamics or switching behaviors.

Monitoring of industrial processes: wind turbines platform. Next, we consider the problem of real-time monitoring industrial processes. Modeling large scale industrial processes is challenging due to the large number of parts involved, nonlinear dynamics and switching behaviors. Switches, in particular, are caused by breaking down of parts (due to wear and tear) and turning processes on and off, which introduce discontinuities in the dynamics. We propose IHYDE as a tool to detect these switches as quickly as possible to prevent lengthy and expensive downtimes in industrial processes.

To put IHYDE to the test, we used real data from a wind turbine platform built in [19]. The data consists of measurements of the current generated by the wind turbine experimental platform under different operating conditions (Table IIC). The system included a 380V power supply, a variable load, a power generator, a motor, a fan, two couplings and a gearbox that transmits the energy generated by the wind wheel to the power generator [19]. We performed experiments under normal and faulty conditions (a broken tooth of gearbox in Supplementary Figure **??**) and down-sampled the measuring current of the wind turbine with a period 0.3 sec. In both experiments, the generator speed was 200 revolutions per minute and the load was 1.5 KNm.

IHYDE was tested under two different scenarios: offline modeling, used, for example, at the design stage; and online modeling, for real-time monitoring. In offline modeling, all the data are available for modeling, while in real-time monitoring only past data are available, and the system is continuously modeled as new data is gathered. In offline modeling, IHYDE identifies two linear subsystems, corresponding to the system in the two different conditions. In addition, it correctly detects the fault right after it happens and infers the transition logic. In online modeling, a model predicts the next time-series data point, and compares it with the real one, when this becomes available. If the difference is high, IHYDE detects a transition, builds a new model, and compares it with the old model to pinpoint the location of the fault. This example focuses on online modeling: the fault is detected within only 3 data points following its occurrence. This application demonstrates the capabilities of IHYDE in online monitoring of industrial processes.

Power systems: smart grids and transmission lines. Smart grids have been gaining considerable attention in the last decades and are transforming how power systems are developed, implemented and operated. They considerably improve efficiency, performance and makes renewable power feasible. In addition, it overhauls aging equipment and facilitates real-time troubleshooting, which decreases brownouts, blackouts, and surges. As with all critical infrastructures, smart grids require strict safety and reliability constraints. Thus, it is of great importance to design monitoring schemes to diagnose anomalies caused by unpredicted or sudden faults [30]. Here, we consider two examples of power systems: real-time modeling to control smart grids and pinpointing the location of a transmission line failure.

We start by illustrating how IHYDE can model and control smart grids in real-time. Accurate model information is not only necessary for daily operation and scheduling, but also critical for other advanced techniques such as state estimation and optimal power flow computation. However, such information is not always available in distribution systems due to frequent model changes [36]. These changes include: high uncertainty in distributed energy resources, such as components being added and removed from the network; unexpected events, such as line faults and unreported line maintenance; and trigger of automatic control and protection measures. We apply IHYDE to identify network models and infer transition logics, capturing model changes from advanced metering infrastructure data and in real-time. The 33-bus benchmark distribution system [25]generates the data. It is a hypothetical 12.66 KV system with a substation, 4 feeders, 32 buses, and 5 tie switches [25]. The system is not well-compensated and lossy, and is widely used to study network reconfiguration problems. Assume the loads on some remote nodes of a feeder suddenly increase, causing voltage sag. Subsequently, an operator takes switch action for load balancing and voltage regulation. Supplementary Figure 10 depicts the switching topologies and the real transition logic. Suppose we can measure all active and reactive power consumption, and voltage phasors of the nodes. Hence, the system is changing between two configurations corresponding to topologies when some switches turn on and off. For each node and subsystem, IHYDE successfully identifies the responding column of the admittance matrices with nearly zero identification errors. The identified admittance matrices at the switching time instants are very different from that of the previous moments, indicating a model switching (corresponding to changes in colors on the data in Table IID). Indeed, the identified logic

is consistent with the real logic and demonstrates that IHYDE can reveal voltage drops at specific nodes in real-time and suggest switch action to avoid sharp voltage drops.

To simulate a transmission line failure, assume a transmission line fails between two buses in the network. We will use a standard benchmark IEEE 14-bus power network (please see [32]). This system consists of generators, transmission lines, transformers, loads and capacitor banks. Looking directly at the generated data (Table IIE), it is not clear when the fault occurred, and much less what happened at the time of failure and where it was located. This is because the power system compensated the failure by rerouting power across other lines. IHYDE, however, can immediately detect the occurrence of this event and determine its location. This is done by estimating the new admittance matrix using only 10 measurements following the failure (corresponding to 166.7 milliseconds, according to the IEEE synchrophasor measurements standard C37.118, 2011). Basically, it successfully discovers both subsystems (normal and failure) from data and calculates the difference of the discovered subsystems (leading to the location of the fault). Given the frequency at which Phasor Measurement Units (PMUs) sample voltage and current, IHYDE is able to locate the fault in a few hundred milliseconds after the event occurs, enabling the operators to detect the event, identify its location, and take remedial actions in real-time.

Medical applications: heart atrial active potential monitoring. The development of medical devices is another active research area. Especially, with the widespread use of wearables and smart devices, there is an exponential growth of data collection. These data requires personalized modeling algorithms to extract critical information for diagnosis and treatments. Within this context, we apply IHYDE to model data gathered from a human atrial action potential (AP) system [26]. The human atrial AP and ionic currents that underlie its morphology are of great importance to our understanding and prediction of the electrical properties of atrial tissues under normal and pathological conditions.

The model captures the spiking of the atrial AP. In particular, two gating variables capture the fast and slow inactivation with switching dynamics. Following a spike, these two variables raise, preventing a new spike. Eventually, as the AP returns to low values, the inactivation dynamics switch back, and in time allow a new spike to take place. The goal is to test whether IHYDE can detect these transitions, together with the rules that led to the switch. Two scenarios are considered here: the first scenario assumes the first-principle model parameterization is available while the second not. In the first scenario,

13

IHYDE indeed identifies the two subsystems, together with their dynamics, and pinpoints the changing logic correctly (Table IIF). For the second scenario, we repeat the modeling of this system, this time assuming that the choice of dictionary functions is unclear and/or the domain knowledge is lacking. In such cases, we consider a canonical dictionary function, such as polynomials approximations. IHYDE can still detect the transition points. However, the nonlinear dynamics are different than the true ones: as expected, it identifies instead a polynomial approximation of the original non-linear dynamics. While these dynamics can still be used for simulation and trajectory prediction, they are not in a form that reveals physical meaning. For an interpretable model, we require domain knowledge. Hence, IHYDE provides a reliable model to study the system and to build devices to detect abnormal AP.

DISCUSSION

This work presents a new algorithm for identifying CPS from data. Current state-of-the-art methods assume either topology of a network, order and/or the exact dynamics of subsystems, number of subsystems, or the switching rules. IHYDE does not require such assumptions. Instead, IHYDE only requires similar assumptions to those in standard system identification. For example, full state measurements, linear dependence of the to-be-identified parameters and the choice of dictionary functions generally guided by the area of the application. IHYDE successfully identifies complex mechanistic models directly from data, including the subsystem dynamics and their associated transition logics. The proposed method differs from classical machine learning tools, such as deep neural network models [5], which typically do not provide insight on the underlying mechanisms of the systems. Typically, the fitted neural network does not provide insight on physical mechanisms of the system, since state-variables and learned parameters have no (direct) meaning. While IHYDE is inspired by prior work in symbolic regression [24], it has much lower computational complexity due to the use of sparse identification and artificial intelligence. As a result, it can solve large-scale CPSs, facilitating its application to complex real-world problems.

After IHYDE models a CPS, the resulting model can help verify the design specifications and predict future trajectories. If the CPS model reveals bugs or flaws in the implementation, it can potentially guide the redesign to achieve the required performance. Applications include robotics and automated vehicles (such as cars and unmanned air/spacecraft), where

data-driven models promise to reduce the reliance on trial and error. Furthermore, IHYDE can monitor, detect, and pinpoint real-time faults of CPSs (for example, power systems), thereby helping avoid catastrophic failures.

As seen in the results section, IHYDE can be applied to a wide range of applications. Supplementary Materials includes additional examples on canonical hybrid dynamical systems [24]. As before, IHYDE successfully identifies both the subsystems and the transition rules with virtually zero error (see Supplementary Example 1 - Example 4).

IHYDE unifies previous results as it can discover not only hybrid dynamical systems, but also non-hybrid dynamical systems (i.e., time-invariant linear and nonlinear systems [10, 22]) as special cases. This was confirmed in Supplementary Materials Section , where IHYDE successfully identified the original canonical dynamical systems from the data in [22] (Supplementary Table 47). Hence, IHYDE provides a unified approach to the discovery of hybrid and non-hybrid dynamical systems.

While the approach has a number of advantages, there are still some open questions. First, it requires a new theory to understand when particular datasets are informative enough to uniquely identify a single (the true) hybrid dynamical system. Identifiability is a central topic in system identification and provides guarantees that there does not exist multiple systems that can produce the same data. This is illustrated in Supplementary Discussions, where we construct several hybrid dynamical systems that yield the exact same data, and hence cannot be differentiated from data alone. A second issue lies in the linear parameterization of the model. For equations whose parameters enter nonlinearly, gradient descent can be applied to obtain a local minimizer, although in this case a global optimum cannot typically be guaranteed. Finally, the choice of dictionary functions is generally guided by the area of the system. Any insight or domain knowledge to construct dictionary functions for hybrid dynamical systems can help reduce computational burden and improve model accuracy. When the domain knowledge in unclear or lacking, canonical dictionary functions, such as polynomials, kernels, Fourier series, can approximate the true dynamics. An example in Supplementary Discussion 3 illustrates how a polynomial series successfully approximates a sinusoid. However, in these cases the exact original true function may be lost or hard to find, as illustrated in Supplementary Example 10. There, while IHYDE can still detect the location of switches, it discovers different dynamics based on the choice of the canonical dictionary function. Nevertheless, these dynamics can still be used for prediction since they

still approximate the main dynamics of each subsystem (see for example, Supplementary Discussion 3).

METHODS

The theoretical foundation of IHYDE algorithm. Motivated by the above example, we shall give a formal definition of hybrid dynamical systems. Physical systems are characterized by inputs $\mathbf{u}(t) \in \mathbb{R}^m$ and outputs $\mathbf{y}(t) \in \mathbb{R}^n$. Based on these variables, at any given time a particular mode $m(t)$ is chosen from a possible total of K modes, i.e., $m(t) \in \{1, 2, ..., K\}$. Each mode corresponds to a particular set of physical parameters. The physical system evolves according to sets of differential equations: $\frac{d\mathbf{y}(t)}{dt} = \mathbf{F}_k\left(\mathbf{y}(t), \mathbf{u}(t)\right), \ k = 1, 2, \ldots, K$, where each $\mathbf{F}_k(\mathbf{y}(t), \mathbf{u}(t))$ is related to the dynamics of subsystem k. Assume $\mathbf{y}(t)$ and $\mathbf{u}(t)$ are sampled at a rate $h > 0$, i.e. sampled at times $0, h, 2h, 3h....$ For fast enough sampling (for small sampling period h), one of the simplest method to approximate derivatives is to consider $\frac{d\mathbf{y}(t)}{dt} \approx \frac{\mathbf{y}(t+h) - \mathbf{y}(t)}{h}$, which yields the discrete-time system $\mathbf{y}(t+h) = \mathbf{y}(t) + h\,\mathbf{F}_k(\mathbf{y}(t), \mathbf{u}(t)) \triangleq \mathbf{f}_k(\mathbf{y}(t), \mathbf{u}(t)), \ k = 1, 2, \ldots, K$. At any given time, the decision of the transition logic to switch to another subsystem is governed by transition rules of the form $m(t+h) = \mathcal{T}(m(t), \mathbf{y}(t), \mathbf{u}(t))$. Hence, the current input-output variables $\mathbf{y}(t), \mathbf{u}(t)$ plus the current subsystem mode $m(t)$ determine, via a function $\mathcal{T}$, the next subsystem mode. Without loss of generality, we can rescale the time variable t so that $h = 1$. Thus, we can construct the following mathematical model for hybrid dynamical systems

$$m(t + 1) = \mathcal{T}(m(t), \mathbf{y}(t), \mathbf{u}(t))$$

$$\mathbf{y}(t + 1) = \mathbf{f}(m(t), \mathbf{y}(t), \mathbf{u}(t)) = \begin{cases} \mathbf{f}_1(\mathbf{y}(t), \mathbf{u}(t)), & \text{if } m(t) = 1 \\ \qquad\qquad \vdots \ , & \qquad\qquad \vdots \\ \mathbf{f}_K(\mathbf{y}(t), \mathbf{u}(t)), & \text{if } m(t) = K. \end{cases}$$

Given the mathematical definition of the hybrid dynamical systems, we can then propose the IHYDE algorithm for discovering such systems from data.

Inferring subsystems. Let $\mathbf{Y}$ and $\mathbf{U}$ denote column vectors containing all the samples of $\mathbf{y}(t)$ and $\mathbf{u}(t)$, respectively, for $t = 1, 2, \ldots, M + 1$, where $M + 1$ is the total number of samples. The first step to identify the subsystems is to construct a library $\mathbf{\Phi}(\mathbf{Y}, \mathbf{U})$ of

nonlinear functions from the input-output data. The exact choice of nonlinear functions in this library depends on the actual application. For example, for simple mechanical systems, $\boldsymbol{\Phi}$ would consist of constant, linear and trigonometric terms; in biological networks, $\boldsymbol{\Phi}$ would consist of polynomial (mass action kinetics) and sigmoidal (enzyme kinetics) terms. Let

$$
\mathbf{Y} = \begin{bmatrix} | & | & | & | & | \\ \mathbf{y}(1) & \mathbf{y}(2) & \ldots & \mathbf{y}(M) \\ | & | & | & | & | \end{bmatrix}^T , \quad \mathbf{U} = \begin{bmatrix} | & | & | & | & | \\ \mathbf{u}(1) & \mathbf{u}(2) & \ldots & \mathbf{u}(M) \\ | & | & | & | & | \end{bmatrix}^T , \quad \bar{\mathbf{Y}} \triangleq \begin{bmatrix} | & | & | & | & | \\ \mathbf{y}(2) & \mathbf{y}(3) & \ldots & \mathbf{y}(M+1) \\ | & | & | & | & | \end{bmatrix}^T .
$$

As an illustration, for polynomials (with $\mathbf{U} = \mathbf{0}$) we would have $\boldsymbol{\Phi}(\mathbf{Y}, \mathbf{U}) = \begin{bmatrix} \mathbf{1} & \mathbf{Y} & \mathbf{Y}^{P_2} & \cdots \end{bmatrix}$ where higher polynomials are denoted as $\mathbf{Y}^{P_2}, \mathbf{Y}^{P_3}$, etc. For instance, $\mathbf{Y}^{P_2}$ denotes quadratic nonlinearities[22]:

$$
\mathbf{Y}^{P_2} = \begin{bmatrix} \mathbf{y}_1^2(1) & \mathbf{y}_1(1)\mathbf{y}_2(1) & \cdots & \mathbf{y}_n^2(1) \\ \mathbf{y}_1^2(2) & \mathbf{y}_1(2)\mathbf{y}_2(2) & \cdots & \mathbf{y}_n^2(2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{y}_1^2(M) & \mathbf{y}_1(M)\mathbf{y}_2(M) & \cdots & \mathbf{y}_n^2(M) \end{bmatrix} .
$$

Basically, each column of $\boldsymbol{\Phi}(\mathbf{Y}, \mathbf{U})$ represents a candidate function for a nonlinearity in $\mathbf{f}_k$. Let the residual as

$$
\mathbf{Z} \triangleq \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 & \ldots & \mathbf{z}_n \end{bmatrix} = \bar{\mathbf{Y}} - \boldsymbol{\Phi}\mathbf{W} - \boldsymbol{\xi},
$$

where $\boldsymbol{\xi}$ is Gaussian noise, i.e., $\boldsymbol{\xi} \sim \mathcal{N}(0, \lambda\mathbf{I})$. The first objective is to find the sparsest possible $\mathbf{Z}$ that fits most input-output data, i.e.,

$$
\mathbf{Z}^* = \arg \min_{\mathbf{Z}} \|\mathbf{Z}\|_{\ell_0}
$$

$$
\text{subject to: } \mathbf{Z} = \bar{\mathbf{Y}} - \boldsymbol{\Phi}\mathbf{W} - \boldsymbol{\xi}.
$$

As a result, the indexes of the zero entries of $\mathbf{Z}^*$ correspond to the indexes for input-output that can be fitted by a single subsystem. This initial idea was an extension of those presented in [23], yet the major difference is that we propose a robust Bayesian algorithm that works for noisy data with performance (See Supplementary Method 1 for comparison).

Assume, without loss of generality, that the dictionary matrix $\boldsymbol{\Phi}$ is full rank. A key step is to define a transformation matrix $\boldsymbol{\Theta}$ in which each column spans the left null space of $\boldsymbol{\Phi}$.

17

Then, it follows that $\boldsymbol{\Theta}\bar{\mathbf{Y}} = \boldsymbol{\Theta}\mathbf{Z} + \boldsymbol{\Theta}\boldsymbol{\xi}$. Let $\tilde{\bar{\mathbf{Y}}} \triangleq \boldsymbol{\Theta}\bar{\mathbf{Y}}$ and $\boldsymbol{\Pi} = \boldsymbol{\Theta}\boldsymbol{\Theta}^T$, then

$$P(\tilde{\bar{\mathbf{Y}}}|\mathbf{Z}) = \mathcal{N}(\tilde{\bar{\mathbf{Y}}}|\boldsymbol{\Theta}\mathbf{Z}, \lambda\boldsymbol{\Pi}) \propto \exp\left[-\frac{1}{2\lambda}\left\|(\tilde{\bar{\mathbf{Y}}} - \boldsymbol{\Theta}\mathbf{Z})^T\boldsymbol{\Pi}^{-1}(\tilde{\bar{\mathbf{Y}}} - \boldsymbol{\Theta}\mathbf{Z})\right\|_F^2\right]. \tag{1}$$

Each column of $\tilde{\bar{\mathbf{Y}}}$ (i.e., $\tilde{\bar{\mathbf{y}}}_i$) can be identified independently for $i = 1, \cdots, n$ (let $\mathbf{z}_i$ be the ith column of $\mathbf{Z}$)

$$P(\tilde{\bar{\mathbf{y}}}_i|\mathbf{z}_i) = \mathcal{N}(\tilde{\bar{\mathbf{y}}}_i|\boldsymbol{\Theta}\mathbf{z}_i, \lambda\boldsymbol{\Pi}) \propto \exp\left[-\frac{1}{2\lambda}(\tilde{\bar{\mathbf{y}}}_i - \boldsymbol{\Theta}\mathbf{z}_i)^T\boldsymbol{\Pi}^{-1}(\tilde{\bar{\mathbf{y}}}_i - \boldsymbol{\Theta}\mathbf{z}_i)\right]. \tag{2}$$

Once we introduce the Gaussian likelihood in (2) and the variational prior

$$\mathcal{N}(\mathbf{z}_i) = \max_{\gamma_j > 0} \prod_j \mathcal{N}(z_{ji}|0, \gamma_j)\varphi(\gamma_j) = \max_{\boldsymbol{\Gamma} \succ \mathbf{0}} \mathcal{N}(\mathbf{z}_i|\mathbf{0}, \boldsymbol{\Gamma}) \prod_j \varphi(\gamma_j),$$

where $\boldsymbol{\Gamma} \triangleq \operatorname{diag}\{\boldsymbol{\gamma}\}$ and $\varphi(\gamma_j)$ is a nonnegative function. The target is to maximize the marginal likelihood as

$$\int \mathcal{N}(\tilde{\bar{\mathbf{y}}}_i|\boldsymbol{\Theta}\mathbf{z}_i, \lambda\boldsymbol{\Pi})\mathcal{N}(\mathbf{z}_i|\mathbf{0}, \boldsymbol{\Gamma}) \prod_j \varphi(\gamma_j)d\mathbf{z}_i. \tag{3}$$

Using results in [10], we can get the following optimization problem jointly on $\mathbf{z}_i$ and $\boldsymbol{\gamma}$,

$$\min_{\mathbf{z}_i, \boldsymbol{\gamma}} \frac{1}{\lambda}\left(\tilde{\bar{\mathbf{y}}}_i - \boldsymbol{\Theta}\mathbf{z}_i\right)^T \boldsymbol{\Pi}^{-1}\left(\tilde{\bar{\mathbf{y}}}_i - \boldsymbol{\Theta}\mathbf{z}_i\right) + \mathbf{z}_i^T\boldsymbol{\Gamma}^{-1}\mathbf{z}_i + \log\det(\lambda\boldsymbol{\Pi} + \boldsymbol{\Theta}\boldsymbol{\Gamma}\boldsymbol{\Theta}^T) + \sum_j \log\varphi(\gamma_j).$$

For the case of uniform priors, let $\varphi(\gamma_j) = 1$. This program can be formulated as a convex-concave procedure (CCCP), i.e., where the first part of the function

$$u(\mathbf{z}_i, \boldsymbol{\gamma}) = \frac{1}{\lambda}\left(\tilde{\bar{\mathbf{y}}}_i - \boldsymbol{\Theta}\mathbf{z}_i\right)^T \boldsymbol{\Pi}^{-1}\left(\tilde{\bar{\mathbf{y}}}_i - \boldsymbol{\Theta}\mathbf{z}_i\right) + \mathbf{z}_i^T\boldsymbol{\Gamma}^{-1}\mathbf{z}_i \tag{4}$$

is convex jointly in $\mathbf{z}_i$ and $\boldsymbol{\gamma}$, and the second part

$$s(\boldsymbol{\gamma}) = \log\det(\lambda\boldsymbol{\Pi} + \boldsymbol{\Theta}\boldsymbol{\Gamma}\boldsymbol{\Theta}^T) \tag{5}$$

is concave in $\boldsymbol{\gamma}$.

Now the high level plan is to optimize over each set of variables iteratively based on CCCP, as follows:

$$\mathbf{z}_i^{k+1} = \arg\min_{\mathbf{z}_i} u(\mathbf{z}_i, \boldsymbol{\gamma}^k),$$

$$\boldsymbol{\gamma}^{k+1} = \arg\min_{\boldsymbol{\gamma} \geq \mathbf{0}} u(\mathbf{z}_i^k, \boldsymbol{\gamma}) + \nabla_{\boldsymbol{\gamma}}s(\boldsymbol{\gamma}^k)^T\boldsymbol{\gamma}. \tag{6}$$

Then we propose our algorithm to solve the above procedure and the pseudo code is summarized in Algorithm 1.

Algorithm 1 Reweighted ℓ_1 type algorithm for Gaussian likelihood

---

1: Initialize the unknown $\mathbf{z}_i$ as a unit vector;

2: Fix $\lambda = 1$ or select $\lambda \in \mathbb{R}^+$ as trail and error which may be empirically helpful;

3: **for** $k = 1, \ldots, K_{\max}$ **do**

4:
$$\mathbf{z}_i^{k+1} = \arg\min_{\mathbf{z}_i} \frac{1}{2} \left(\tilde{\mathbf{y}}_i - \boldsymbol{\Theta}\mathbf{z}_i\right)^T \boldsymbol{\Pi}^{-1} \left(\tilde{\mathbf{y}}_i - \boldsymbol{\Theta}\mathbf{z}_i\right) + \lambda \sum_j |\alpha_j^k \cdot z_{ji}|; \qquad (7)$$

5: $\quad \gamma_j^{k+1} = \left|\dfrac{z_{ji}^{k+1}}{\alpha_j^k}\right|,\ \alpha_j^{k+1} = \left(\boldsymbol{\theta}_j^T (\lambda\boldsymbol{\Pi} + \boldsymbol{\Theta}\boldsymbol{\Gamma}^{k+1}\boldsymbol{\Theta}^T)^{-1}\boldsymbol{\theta}_j\right)^{\frac{1}{2}};$

6: $\quad$ **if** a stopping criterion is satisfied **then**

7: $\qquad$ Break.

8: $\quad$ **end if**

9: **end for**

This step identifies which time points correspond to which subsystem. The second objective identifies the actual dynamics of each subsystem. The algorithm starts with subsystem k, which is the one associated with the largest number of time points. Let seq denote those time points associated with subsystem k. Once every data point is associated to different subsystems, next, we shall infer the dynamics of every subsystem. We set up yet another sparse regression problem to determine the sparse vectors of coefficients. The sparse coefficients $\mathbf{w}_k^*$ of subsystem k are then identified by solving the following optimization problem

$$\mathbf{w}_k^* = \arg\min_{\mathbf{w}_k} \frac{1}{2} \|\bar{\mathbf{Y}}[\text{seq}, k] - \boldsymbol{\Phi}[\text{seq}, :]\mathbf{w}_k\|_{\ell_2}^2 + \lambda_{\mathbf{w}} \|\mathbf{w}_k\|_{\ell_1}.$$

where $\lambda_{\mathbf{w}}$ is a hyperparameter that trades off estimation error and model complexity. These hyperparameters are principally tuned using results in Supplementary Method 1. The algorithm removes the time points seq and repeats these two steps iteratively until all subsystems have been identified and no data is left. Further details are found in Supplementary Algorithm 2.

Inferring transition logics. Once every data point has been classified to different subsystems, define $\eta_i(t)$ as the set membership: it equals to 1 only if the subsystem i is active at discrete-time t, otherwise it equals to 0. These functions are known from the information in the subsystem identification above. Here, we are interested in learning what functions trigger the switch from one subsystem to another. Define also $\text{step}(x)$, which equals to 1 if $x \geq 0$, and 0 otherwise. Mathematically, we are searching for a nonlinear function

g, such that step($g(\mathbf{y}(t), \mathbf{u}(t))$) specifies the membership. Due to non-differentiability of step functions at 0, we alternatively relax the step function to a sigmoid function, i.e., $\eta_j(t+1) \approx \frac{1}{1+e^{-g(\mathbf{y}(t),\mathbf{u}(t))}}$ [24], where j is a potential subsystem that can jump to at time $t+1$. Assuming we are in subsystem i at time t, the fitness function to jump to subsystem j at time $t+1$ is then we solve the following optimization problem

$$\min_g \sum_{t=1}^{M-1} \eta_i(t) \left\| \eta_j(t+1) - \frac{1}{1+e^{-g(\mathbf{y}(t),\mathbf{u}(t))}} \right\|_{\ell_2}^2. \tag{8}$$

To solve the optimization problem in Eq. (8), we can parameterize $g(\mathbf{y}(t), \mathbf{u}(t))$ as a linear combination of over-determined dictionary matrix, i.e., $g(\mathbf{y}(t), \mathbf{u}(t)) \triangleq \mathbf{\Psi}(\mathbf{Y}, \mathbf{U})\mathbf{v}$, in which $\mathbf{\Psi}$ can be constructed similarly to $\mathbf{\Phi}$ in the previous subsection and $\mathbf{v}$ is a vector of to-be-discovered parameters. Further details can be found in Supplementary Algorithm 3. It should be noted that, the optimization problem in Eq. (8) is also convex, which yields a computationally efficient solution.

Data availability. All data needed to evaluate the conclusions in the paper are available at [https://github.com/HAIRLAB/CPSid] except datasets from [24] (Example 1 to 4 in the Supplementary Information). **Code availability.** The code implementation is available at [https://github.com/HAIRLAB/CPSid].

---

[1] Poovendran, R. Cyber-physical systems: Close encounters between two parallel worlds [point of view]. *Proc. IEEE* **98**, 1363-1366 (2010).

[2] Antsaklis, P. A Brief Introduction to the Theory and Applications of Hybrid Systems. *Proc. IEEE* **88**, 879-887 (2000).

[3] Aihara, K. & H. Suzuki, Theory of hybrid dynamical systems and its applications to biological and medical systems. *Philos. Trans. R. Soc. A-Math. Phys. Eng. Sci.* **368**, 4893-4914 (2010).

[4] Wooden, D., Powers, M., Egerstedt, M., Christensen,H. & Balch, T. A modular, hybrid system architecture for autonomous, urban driving. *J. Aerosp. Inf. Syst.* **4**, 1047-1058 (2012).

[5] Lecun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436-444 (2015).

[6] Kutz, J. N. *Data-driven modeling and scientific computation: methods for complex systems and big data* (Oxford University Press, 2013).

[7] Wang, W. X., Lai, Y. C. & Grebogi, C. Data based identification and prediction of nonlinear and complex dynamical systems. *Phys. Rep.* **644**, 1-76 (2016).

[8] Schaft, A. van der & Schumacher, J. *An Introduction to Hybrid Dynamical Systems* (Springer-Verlag, London, 2000).

[9] Schmidt, M. & Lipson, H. Distilling free-form natural laws from experimental data. *Science* **324**, 81-85 (2009).

[10] Pan, W., Yuan, Y., Goncalves, J. & Stan, G. B. Reconstruction of arbitrary biochemical reaction networks: A compressive sensing approach. In *Proceedings of the 51st IEEE Conference on Decision and Control*, 2334-2339 (2012).

[11] Wang, W. X., Yang, R., Lai, Y. C., Kovanis, V. & Grebogi, C. Predicting catastrophes in nonlinear dynamical systems by compressive sensing. *Phys. Rev. Lett.* **106**, 154101 (2011).

[12] Chang, Y. H. & Tomlin, C. Data-driven graph reconstruction using compressive sensing. In *Proceedings of the 51st IEEE Conference on Decision and Control*, 1035-1040 (2012).

[13] Brunton, S. L., Proctor, J. L. & Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl. Acad. Sci. USA* **113**, 3932-3937 (2016).

[14] Ohlsson, H. & Ljung, L. Identification of switched linear regression models using sum-of-norms regularization. *Automatica* **49**, 1045-1050 (2013).

[15] Paoletti, S., Juloski,A. L., Ferrari-Trecate, G. & Vidal, R. Identification of hybrid systems a tutorial. *Eur. J. Control* **13**, 242-260 (2007).

[16] Vidal,R., Soatto,S., Ma, Y. & Sastry, S. An algebraic geometric approach to the identification of a class of linear hybrid systems. In *Proceedings of the IEEE Conference on Decision and Control* , 167-172 (2003).

[17] Bako, L. Identification of switched linear systems via sparse optimization. *Automatica* **47**, 668-677 (2011).

[18] Ozay, N., Sznaier, M., Lagoa, C. & Camps, O. A sparsification approach to set membership identification of a class of affine hybrid systems. In Proceedings of the IEEE Conference on Decision and Control, 123-130 (2008).

[19] Roll, J., Bemporad, A. & Ljung, L. Identification of piecewise affine systems via mixed-integer programming. *Automatica* **40**, 37-50 (2004).

[20] Bemporad, A., Garulli, A., Paoletti, S. & Vicino, A. A bounded-error approach to piecewise

affine system identification. *IEEE Trans. Autom. Control* **50**, 1567-1580 (2005).

[21] Juloski, A. L., Weiland, S. & Heemels, W. A Bayesian approach to identification of hybrid systems. *IEEE Trans. Autom. Control* **50**,1520-1533 (2005).

[22] Nakada, H., Takaba, K. & Katayama, T. Identification of piecewise affine systems based on statistical clustering technique. *Automatica* **41**, 905-913 (2005).

[23] Ferrari-Trecate, G., Muselli, M., Liberati,D. & Morari, M. A clustering technique for the identification of piecewise affine systems. *Automatica* **39**, 205-217 (2003).

[24] Oishi, M. & May, E. Addressing biological circuit simulation accuracy: Reachability for parameter identification and initial conditions. In *Proceedings of the IEEE-NIH Life Science Systems and Applications Workshop*, 152-155 (2007).

[25] Thai, J. & Bayen, A. M. State estimation for polyhedral hybrid systems and applications to the Godunov scheme for highway traffic estimation. *IEEE Trans. Autom. Control* **60**, 311-326 (2015).

[26] Candes, E. J. Compressive sampling. In *Proceedings of the international congress of mathematicians*, 1433-1452 (2006).

[27] Ljung, L. *System identification: theory for the user* (PTR Prentice Hall, Upper Saddle River, NJ 1999).

[28] Chua, L. O., Itoh, M., Kocarev, L.& Eckert, K. Chaos synchronization in Chua's circuit. *J. Circuits Syst. Comput.* **2**, 705-708 (2011).

[29] He, Q., Guo, Y., Wang, X., Ren, Z. & Li, J. Gearbox fault diagnosis based on RB-SSD and MCKD. *China Mechanical Engineering* **28**, 1528-1534 (2017).

[30] Pan, W., Yuan, Y., Sandberg, H., Goncalves, J. & Stan, G. B. Online fault diagnosis for nonlinear power systems. *Automatica* **55**, 27-36 (2015).

[31] Baran, M. & Wu, F. Network reconfiguration in distribution systems for loss reduction and load balancing. *IEEE Trans. Power Deliv.* **4**, 1401-1407 (1989).

[32] Christie, P. D. Power Systems Test Case Archive. Seattle, WA, USA: Univ. Washington, 1999. [Online]. Available: `https://www2.ee.washington.edu/research/pstca/pf14/pg\_tca14bus.htm`.

[33] Courtemanche, M., Ramirez, R. & Nattel, S. Ionic mechanisms underlying human atrial action potential properties: insights from a mathematical model. *Am. J. Physiol.-Heart Circul. Physiol.* **275**, 301-321 (1998).

[34] Ly, D. L. & Lipson, H. Learning symbolic representations of hybrid dynamical systems. *J. Mach. Learn. Res.* **13**, 3585-3618 (2012).

[35] Lygeros, J. Tomlin, C. & Sastry, S. *Hybrid Systems: Modeling, Analysis and Control* (UC Berkeley / ETH Zurich lecture notes, 2008)

[36] Weng, Y., Liao, Y. & Rajagopal, R. Distributed energy resources topology identification via graphical modeling, *IEEE Trans. Power Syst.* **32**, 2682-2694 (2017).

[37] Breschi, V., Piga, D. & Bemporad, A. Piecewise affine regression via recursive multiple least squares and multicategory discrimination. *Automatica* **73**, 155-162 (2016).

[38] Hartmann, A., Lemos, J. M., Costa, R. S., Xavier, J. & Vinga, S. Identification of switched ARX models via convex optimization and expectation maximization. *J. Process Control* **28**, 9-16 (2015).

[39] Dinh, T. P., Le, H. M., Le Thi, H. A. & Lauer, F. A difference of convex functions algorithm for switched linear regression, *IEEE Trans. Autom. Control* **59**, 2277-2282 (2014).

[40] Ozay, N., Sznaier, M., Lagoa, C. M. & Camps, O. I. A sparsification approach to set membership identification of switched affine systems, *IEEE Trans. Autom. Control*, **57**, 634-648 (2011).

[41] Pillonetto, G. A new kernel-based approach to hybrid system identification. *Automatica* **70**, 21-31 (2016).

ADDITIONAL INFORMATION

Supplementary Information: Data Driven Discovery of Cyber Physical Systems

Ye Yuan et al.

SUPPLEMENTARY FIGURES
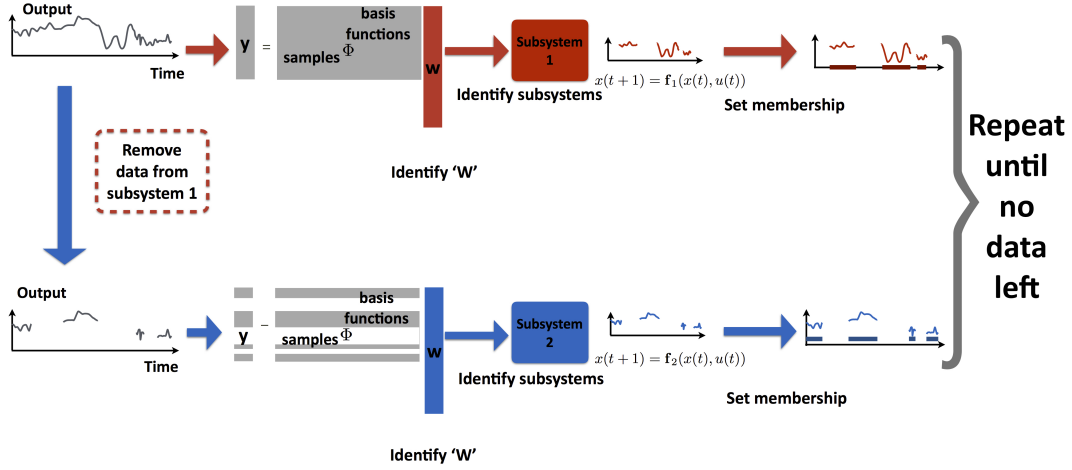
Supplementary Figure 1. **Schematics of the proposed subsystems identification algorithm.** We construct a library of nonlinear functions $\mathbf{\Phi}$. We formulate an iterative convex optimization method to infer the number of subsystems and the underlying system models for every subsystem. More specifically, we first identify a best model that fits the majority of data, then we remove the fitted data and re-do the identification until no data are left.
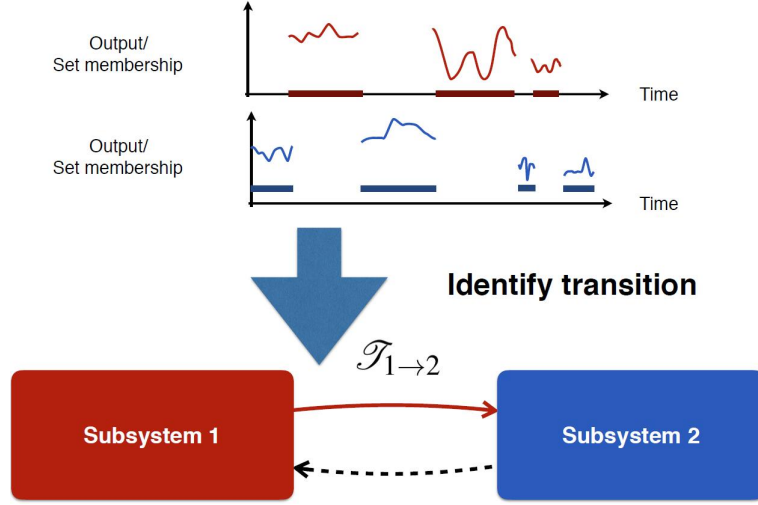
Supplementary Figure 2. **Illustration of the proposed Algorithm to identify transition logic.** Using the membership of every classified data point, we apply logistic regression to infer the logic between every pair of identified subsystems, i.e., $\mathcal{T}_{i \to i'}$ for every i and i'.

(a)

Subsystem 1
$y = 1$

$u > 0.5$

Subsystem 2
$y = -1$

$u < -0.5$

(b)

Subsystem 1
$y = 0.5u^2$
$+u - 0.5$

$u > 0.98$

Subsystem 2
$y = -0.5u^2$
$+u + 0.5$

$u < -0.98$

(c)

$u_4 = 1$

Subsystem 1
$y = u_2 - u_1$

$u_3 = 1$

Subsystem 2
$y = \dfrac{1}{u_1 - u_2}$

$u_5 = 1$

$u_4 = 1$

$u_5 = 1$

$u_3 = 1$

Subsystem 3
$y = 0$

(d)

$u_1^2 + u_2^2 > 25$

Subsystem 1
$y = u_1 u_2$

Subsystem 2
$y = \dfrac{6u_1}{u_2 + 6}$

$u_1^2 + u_2^2 < 9$

$u_1 u_2 < 0$

Subsystem 3
$y = \dfrac{u_1 + u_2}{u_1 - u_2}$
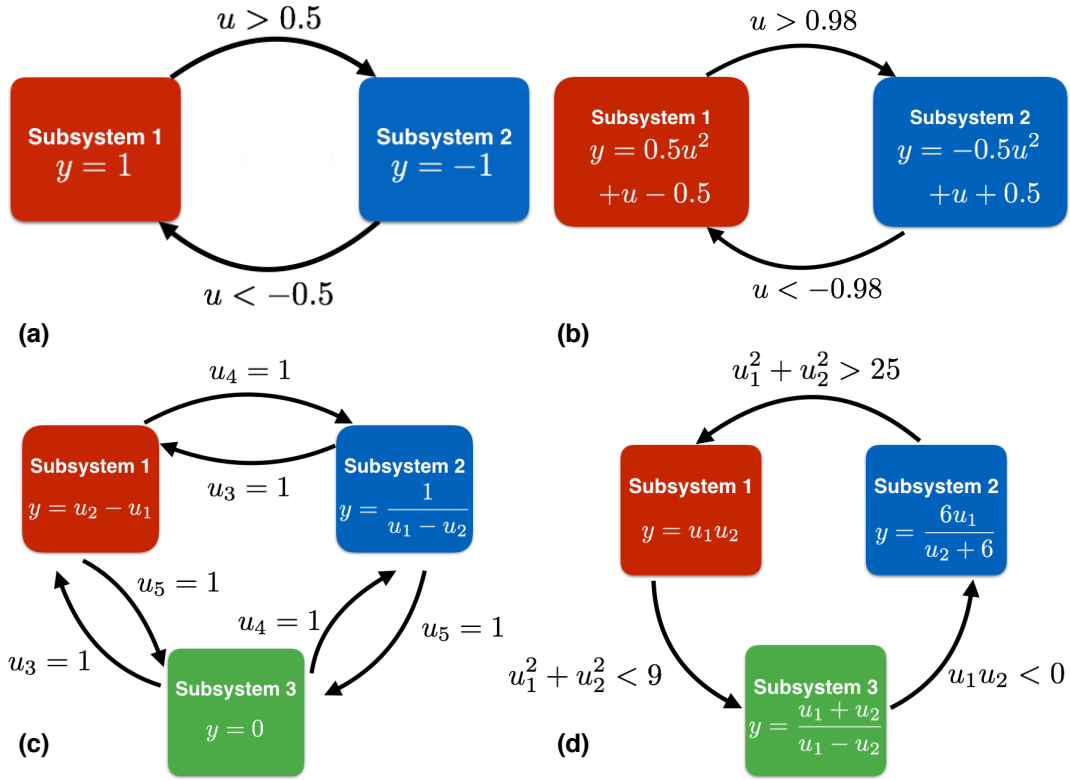
Supplementary Figure 3. The hybrid dynamical system model with the measured input-output u and y. (a) Hysteresis Relay system. (b) Continuous Hysteresis Loop system. (c) Phototaxis Robot system. (d) The nonlinear hybrid dynamical system.

$r(t)$ $+$ $e(t)$ $\quad$ $u(t)$ $\quad$ $c(t)$

$D(s)$ $\quad$ $Object$

$-$
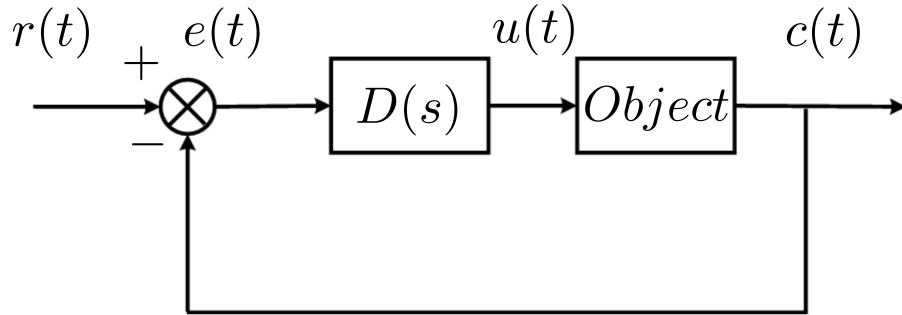
Supplementary Figure 4. The position PI controller structure for the autonomous car.

3

Implemented System

straight$(k) = 1$
straight$(k-1) = 1$

Subsystem 1
$\Delta u(k) = P(v(k-1)$
$-v(k)) + I(v_{\text{expect1}}$
$-v(k))$

straight$(k) = 0$
straight$(k-1) = 1$

Subsystem 3
$\Delta u(k) = P(v(k-1) - v(k))$
$+P(v_{\text{expect1}} - v_{\text{expect2}})$
$+I(v_{\text{expect1}} - v(k))$

Subsystem 4
$\Delta u(k) = P(v(k-1) - v(k))$
$+P(v_{\text{expect2}} - v_{\text{expect1}})$
$+I(v_{\text{expect2}} - v(k))$

straight$(k) = 1$
straight$(k-1) = 0$

Subsystem 2
$\Delta u(k) = P(v(k-1)$
$-v(k)) + I(v_{\text{expect2}}$
$-v(k))$

straight$(k) = 0$
straight$(k-1) = 0$

Designed System

Subsystem 1
$\Delta u(k) = P(v(k-1)$
$-v(k)) + I(v_{\text{expect1}}$
$-v(k))$

straight$(k) = 1$

straight$(k) = 0$

Subsystem 2
$\Delta u(k) = P(v(k-1)$
$-v(k)) + I(v_{\text{expect2}}$
$-v(k))$

Supplementary Figure 5. Left: a more complicated hybrid dynamical system model due to discretization and switching. Right: the correct hybrid dynamical system model that we would like to design.
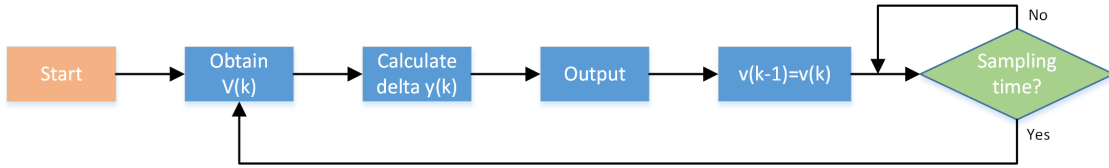
Supplementary Figure 6. The flow chart of the PI algorithm.

Implemented System   **Designed System**

Subsystem 1
$$\Delta u(k) = P[v(k-1) - v(k)] + I(v_{\text{expect2}} - v(t))$$

Subsystem 1
$$\Delta u(k) = P[v(k-1) - v(k)] + I(v_{\text{expect1}} - v(t))$$

$straight = 1$

$straight = 1$

Subsystem 2
$$\Delta u(k) = P[v(k-1) - v(k)] + I(v_{\text{expect1}} - v(t))$$

$straight = 0$

Subsystem 2
$$\Delta u(k) = P[v(k-1) - v(k)] + I(v_{\text{expect2}} - v(t))$$

$straight = 0$
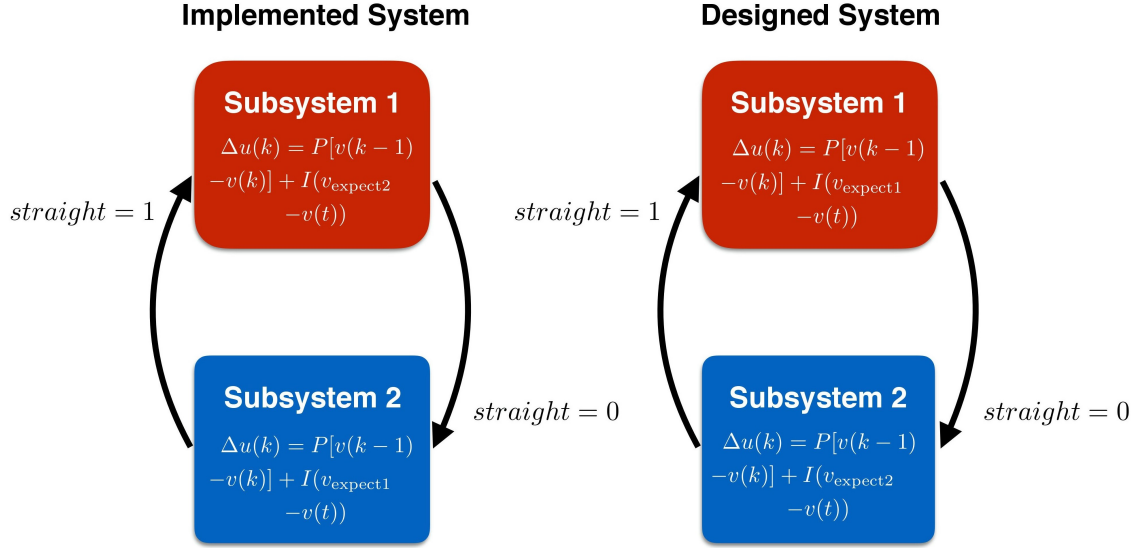
Supplementary Figure 7. The IHYDE pinpoints the implementation error that leads to a failure in the autonomous car experiment. Left: the identified model from experimental data. Right: the designed system. The two subsystems are swapped around due to a design bug.

(a)

(b)

(c)

(d)
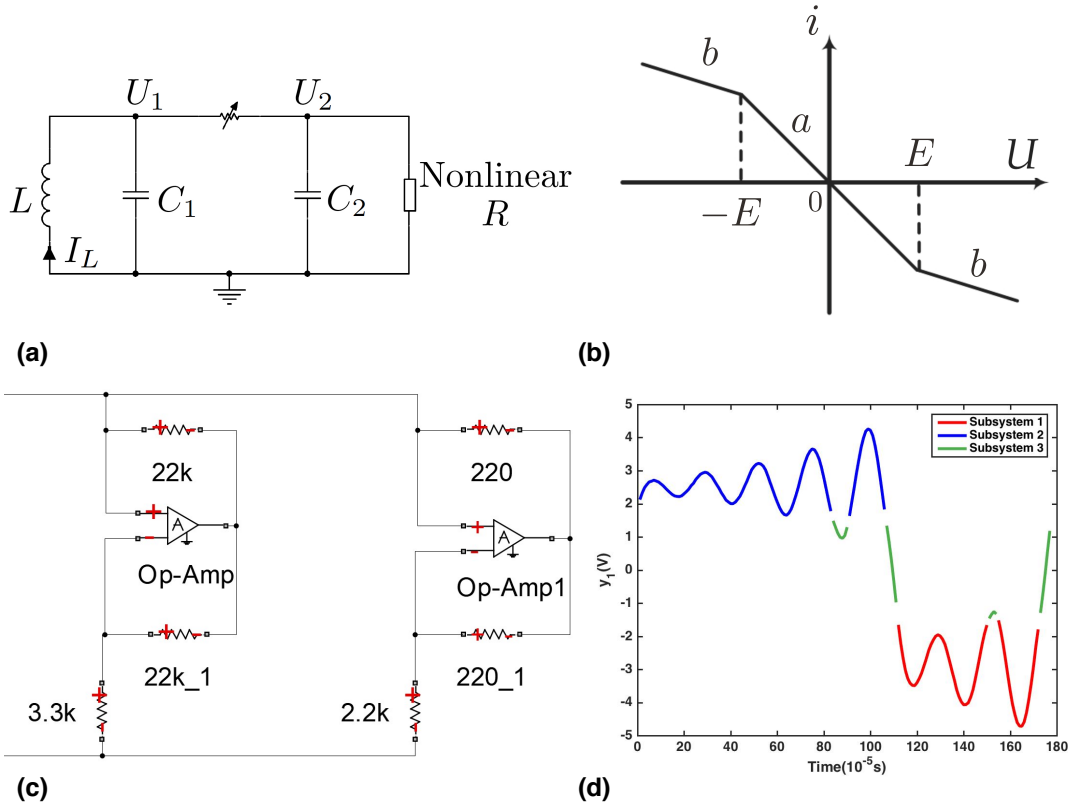
Supplementary Figure 8. Experiments of Chua's circuit. (a) the circuit structure. (b) the current-voltage characteristics of the nonlinear resistor. (c) the circuit structure of nonlinear resistor with specified current-voltage realization. (d) The output trajectory associated with different colors generated by different subsystems.
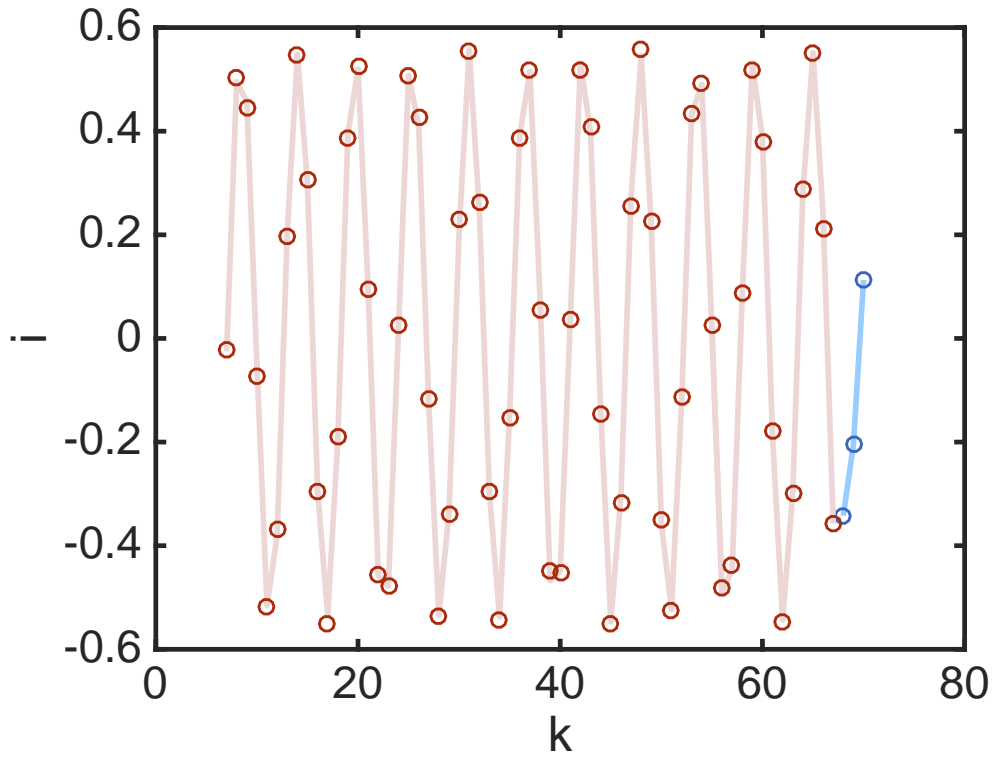
Supplementary Figure 9. The data fitting curve using data obtained from the wind turbine platform using IHYDE.The original time-series data (lines connecting the dots) in the color associated with the subsystem, the fitted data from the identified models (dots), and the location of the transitions (changes in colors).
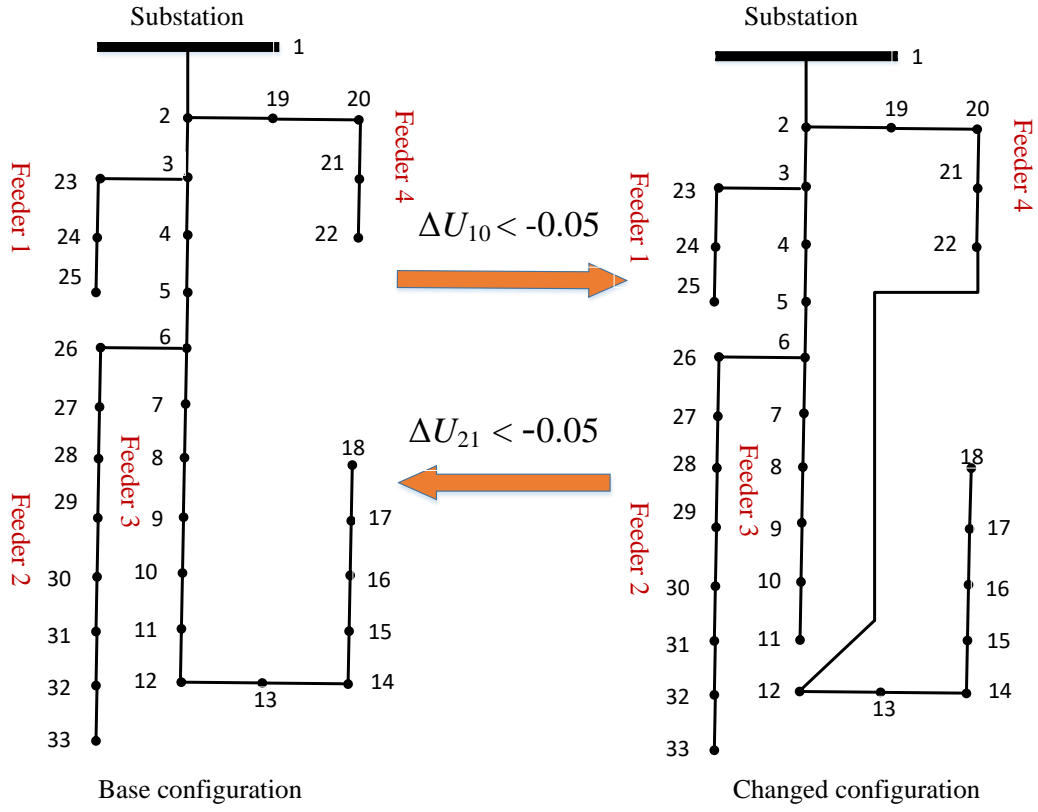
Supplementary Figure 10. Subsystem models and transition logic of the smart grid example.
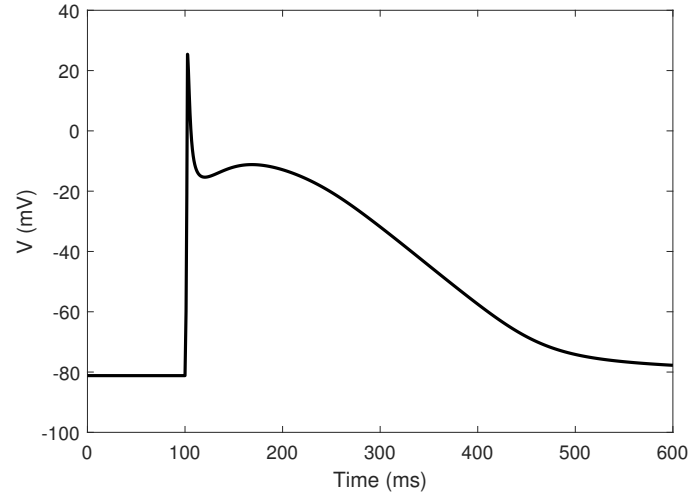
Supplementary Figure 11. Model action potential V during stimulation at the frequency of 1 Hz.
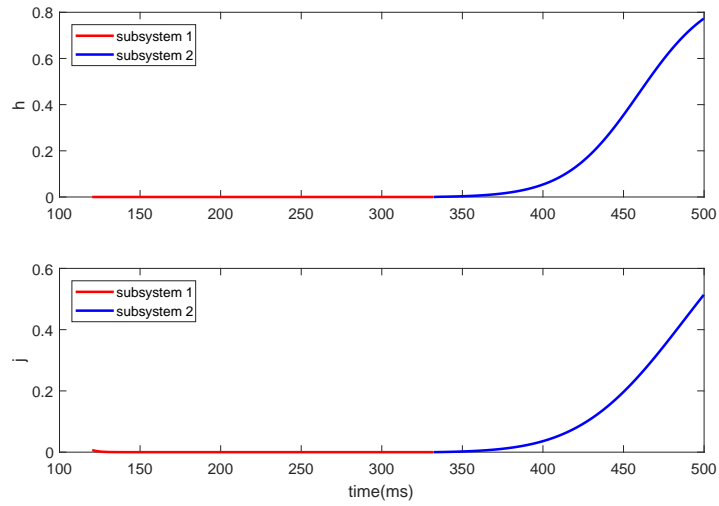
Supplementary Figure 12. The value of gating variable h and j. Different colors denote data that are produced from different subsystems.
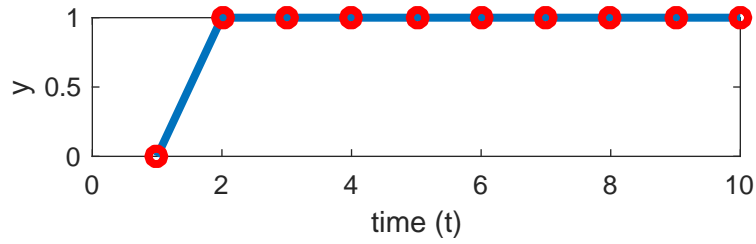
Supplementary Figure 13. The observed output of Eq. (37) when a Dirac delta function is applied to stimulate the system.
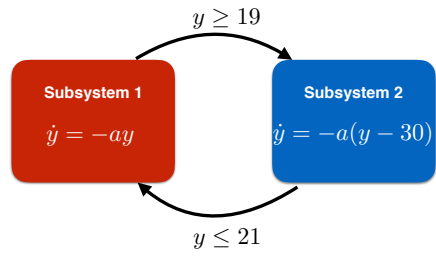
Supplementary Figure 14. A counterexample of a constructed hybrid dynamical system that is not able to be identified from data.

SUPPLEMENTARY TABLES

Supplementary Table 1. The identified results of method in [23] and IHYDE

Metrics	Iteration=1		Iteration=5
	Method [23]	IHYDE	IHYDE
parameters	$\lambda = 0.015,$ $\epsilon = 0.1$	$\lambda_{\mathbf{z}} = 0.015, \epsilon_{\mathbf{z}} = 0.25,$ $\lambda_{\mathbf{w}} = 0.015, \epsilon_{\mathbf{w}} = 0.2$	$\lambda_{\mathbf{z}} = 0.03, \epsilon_{\mathbf{z}} = 1e-4,$ $\lambda_{\mathbf{w}} = 0.008, \epsilon_{\mathbf{w}} = 0.1$
Number of Systems	2	2	2
Number of misclassified points	96	75	41
Dictionary	$1 \quad u \quad u^2 \quad u^3 \quad u^4 \quad u^5$		
Actual Subsystems 1	$y = 0.5u^2 + u - 0.5$		
Identified Subsystems 1	$y = 1.0371u^5 - 0.1238u^4$ $-0.6507u^3 + 0.5462u^2$ $+1.0627u - 0.5005$	$y = 0.0753u^4 + 0.3866u^2$ $+1.0079u - 0.4844$	$y = 0.4841u^2$ $+0.9978u - 0.4984$
Actual Subsystems 2	$y = -0.5u^2 + u + 0.5$		
Identified Subsystems 2	$y = -0.6749u^5 + 0.2908u^4$ $+0.8616u^3 - 0.7386u^2$ $+0.8132u + 0.4322$	$y = -0.5417u^2$ $+1.0613u + 0.4829$	$y = -0.4806u^2$ $+0.9995u + 0.4882$

Supplementary Table 2. The identified subsystems and the selected hyperparameters based on the minimum error principle.

Data Set	$\lambda_{\mathbf{z}}$	$\lambda_{\mathbf{w}}$	$\epsilon_{\mathbf{z}}$	$\epsilon_{\mathbf{w}}$	Mode (m_k)	Actual subsystem	Identified subsystem	Dictionary
Hysteresis Relay	0.1	0.01	$1e-4$	0.0224	1	$y = 1$	$y = 1.0020$	polynomials in u up to 5th order
					2	$y = -1$	$y = -1.0014$	
Continuous Hysteresis Loop	0.1	0.1	$1e-4$	0.1184	1	$y = 0.5u^2 + u - 0.5$	$y = 0.4275u^2 + 0.9954u - 0.4802$	polynomials in u up to 5th order
					2	$y = -0.5u^2 + u + 0.5$	$y = -0.5226u^2 + 1.0190u + 0.4999$	
Phototaxis Robot	$1e-3$	0.1	$1e-4$	0.1619	1	$y = u_2 - u_1$	$y = 0.9947u_2 - 0.9947u_1$	$1, u_1 - u_2,$ $\frac{1}{u_1-u_2}, u_1^2, u_2^2$
					2	$y = \frac{1}{u_1-u_2}$	$y = \frac{0.9707}{u_1-u_2}$	
					3	$y = 0$	$y = 0.0062u_1 - 0.0062u_2$	
Non-linear-Hybrid-System	$1e-4$	0.01	$1e-4$	1.2036	1	$y = u_1u_2$	$y = 0.9951u_1u_2$	$\frac{u_1+u_2}{u_1-u_2}, \frac{u_1}{6+u_2}, u_1u_2,$ $u_1, u_2, \sin(u_1),$ $\sin(u_2), u_1^2, u_2^2$
					2	$y = \frac{6u_1}{6+u_2}$	$y = \frac{5.9567u_1}{6+u_2}$	
					3	$y = \frac{u_1+u_2}{u_1-u_2}$	$y = 0.9958\frac{u_1+u_2}{u_1-u_2}$	

Supplementary Table 3. A summary of datasets used for IHYDE.

Data Set	Mode (m_k)	Behavior	No. of Destination Points	No. of Destination Mode	Transition	No. of Transitions
Hysteresis Relay	1	y=1	1004	2	$u > 0.5$	33
	2	y=-1	996	1	$u < -0.5$	32
Continuous Hysteresis Loop	1	$y = 0.5u^2 + u - 0.5$	999	2	$u > 0.98$	21
	2	$y = -0.5u^2 + u + 0.5$	1001	1	$u < -0.98$	21
Phototaxis Robot	1	$y = u_2 - u_1$	654	2	$u_4 = 1$	10
				3	$u_5 = 1$	18
	2	$y = \frac{1}{u_1 - u_2}$	585	1	$u_3 = 1$	14
				3	$u_5 = 1$	11
	3	$y = 0$	761	1	$u_3 = 1$	15
				2	$u_4 = 1$	14
Non-linear Hybrid System	1	$y = u_1 u_2$	605	3	$u_1^2 + u_2^2 < 9$	157
	2	$y = \frac{6u_1}{6+u_2}$	738	1	$u_1^2 + u_2^2 > 25$	158
	3	$y = \frac{u_1+u_2}{u_1-u_2}$	657	1	$u_1 u_2 < 0$	157
Power Grid Fault Detection	1	Normal	30	2	$t > 30$	1
	2	Fault	10	1		0
Smart Grid	1	Topology1	180	2	$\Delta|U_{10}| < -0.05$	3
	2	Topology2	180	1	$\Delta|U_{21}| < -0.05$	2
Wind turbine (experimental data)	1	Normal	61	2	$k > 67$	1
	2	Fault	3			0
Chua's Circuit (experimental data)	1	$10^{-5}\frac{dy_1}{dt} = 1.0858y_2$ $-0.2115y_1 - 0.5349$	57	2	$y_1 > -1.5$	2
	2	$10^{-5}\frac{dy_1}{dt} = 1.0858y_2$ $+0.1451y_1$	21	1	$y_1 < -1.5$	2
				3	$y_1 > 1.5$	1
	3	$10^{-5}\frac{dy_1}{dt} = 1.0858y_2$ $-0.1994y_1 + 0.5168$	99	2	$y_1 < 1.5$	2
Autonomous Car (experimental data)	1	$\Delta u(k) = 9.5(380 - v(k))$ $+48(v(k-1) - v(k))$	147	2	$straight = 0$	2
	2	$\Delta u(k) = 9.5(280 - v(k))$ $+48(v(k-1) - v(k))$	249	1	$straight = 1$	2

Supplementary Table 4. The detailed information of Hysteresis Relay data.

Original data points	2000
Sampling rate	1
Used data points	2000
Number of points (subsystem 1)	1004
Number of points (subsystem 2)	996
Number of transitions (S1 to S2)	33
Number of transitions (S2 to S1)	32

Supplementary Table 5. The identified result and details of Hysteresis Relay.

Noise	$N_p = 0$	$N_p = 2\%$	$N_p = 4\%$	$N_p = 6\%$
Library $\mathbf{\Phi}$		1		
Identified subsystem 1	$y = 0.9999$	$y = 1.0013$	$y = 1.0027$	$y = 1.0040$
Identified subsystem 2	$y = -0.9999$	$y = -1.0004$	$y = -1.0009$	$y = -1.0014$
$\lambda_{\mathbf{z}}$		$1e-3$		
$\epsilon_{\mathbf{z}}$		$1e-4$		
$\lambda_{\mathbf{w}}$		0.05		
$\epsilon_{\mathbf{w}}$		0.2		
Number of misclassified points		0		

14

Supplementary Table 6. The identified result and details of Hysteresis Relay with redundant dictionary functions.

Noise	$N_p = 0$	$N_p = 2\%$	$N_p = 4\%$	$N_p = 6\%$
Library $\mathbf{\Phi}$	$1\ u\ u^2\ u^3\ u^4\ u^5$			
Identified subsystem 1	$y = 0.9999$	$y = 1.0013$	$y = 1.0027$	$y = 1.0038$
Identified subsystem 2	$y = -0.9999$	$y = -1.0004$	$y = -1.0009$	$y = -1.0014$
$\lambda_{\mathbf{z}}$	$1e - 3$			
$\epsilon_{\mathbf{z}}$	$1e - 4$			
$\lambda_{\mathbf{w}}$	0.05			
$\epsilon_{\mathbf{w}}$	0.2			
Number of misclassified points	0			

Supplementary Table 7. The identified transition logic of Hysteresis Relay.

Systems	Subsystem 1	Subsystem 2
Subsystem 1		$u > 0.4995$
Subsystem 2	$u < -0.4987$	
Library $\mathbf{\Psi}$	$1\quad u$	
β	0.1	

Supplementary Table 8. The identified transition logic of Hysteresis Relay when existing redundant dictionary functions.

Systems	Subsystem 1	Subsystem 2				
Subsystem 1		$u > 0.4995$				
Subsystem 2	$u < -0.4987$					
Library $\boldsymbol{\Psi}$	1	u	$e^{(10(\sin(u^2))+10)}$	$\frac{\log(|u|)}{\sin(u)}$	u^2	u^4
β		0.1				

Supplementary Table 9. The detailed information of Continuous Hysteresis Loop data.

Original data points	2000
Sampling rate	1
Used data points	2000
Number of points (subsystem 1)	999
Number of points (subsystem 2)	1001
Number of transitions (S1 to S2)	21
Number of transitions (S2 to S1)	21

Supplementary Table 10. The identified systems of Continuous Hysteresis Loop for both noiseless and noisy datasets. We also present the tuning parameters for different noise levels.

Noise	$N_p = 0$	$N_p = 2\%$	$N_p = 4\%$	$N_p = 6\%$
Library $\mathbf{\Phi}$	polynomials in u up to second order			
Identified subsystem 1	$y = 0.4998u^2$ $+1.0000u$ -0.4999	$y = 0.4989u^2$ $+1.0003u$ -0.4996	$y = 0.4920u^2$ $+0.9999u$ -0.4982	$y = 0.4842u^2$ $+0.9980u$ -0.4961
Identified subsystem 2	$y = -0.5042u^2$ $+1.0021u$ $+0.5008$	$y = -0.5072u^2$ $+1.0031u$ $+0.5015$	$y = -0.5172u^2$ $+1.0055u$ $+0.5032$	$y = -0.5133u^2$ $+0.9966u$ $+0.5027$
$\lambda_{\mathbf{z}}$	0.005	0.005	0.008	0.03
$\epsilon_{\mathbf{z}}$	$1e-4$			
$\lambda_{\mathbf{w}}$	0.005	0.001	0.005	0.008
$\epsilon_{\mathbf{w}}$	0.04	0.04	0.08	0.105
Number of misclassified points	0	17	45	88

Supplementary Table 11. The identified result and details of Continuous Hysteresis Loop with redundant dictionary functions.

Noise	$N_p = 0$	$N_p = 2\%$	$N_p = 4\%$	$N_p = 6\%$
Library $\mathbf{\Phi}$		$1 \; u \; u^2 \; e^{-u} \; \frac{u^3}{e^u} \; \frac{\cos(2u)}{\sin(u)^3}$		
Identified subsystem 1	$y = 0.4999u^2$ $+1.0000u$ -0.4999	$y = 0.5001u^2$ $+1.0001u$ -0.4998	$y = 0.4919u^2$ $+0.9995u$ -0.4979	$y = 0.4811u^2$ $+0.9994u$ -0.4956
Identified subsystem 2	$y = -0.5010u^2$ $+0.9995u$ $+0.5002$	$y = -0.4979u^2$ $+0.9990u$ $+0.5001$	$y = -0.5123u^2$ $+1.0000u$ $+0.5034$	$y = -0.5275u^2$ $+0.9999u$ $+0.5047$
$\lambda_{\mathbf{z}}$	0.005	0.005	0.008	0.03
$\epsilon_{\mathbf{z}}$		$1e-4$		
$\lambda_{\mathbf{w}}$	0.005	0.001	0.005	0.008
$\epsilon_{\mathbf{w}}$	0.04	0.04	0.08	0.105
Number of misclassified points	0	11	45	94

Supplementary Table 12. The identified transition logic of the Continuous Hysteresis Loop with noiseless data and redundant dictionary functions.

System	Subsystem 1	Subsystem 2
Subsystem 1		$u > 0.9803$
Subsystem 2	$u < -0.9799$	
Library $\mathbf{\Psi}$	$1 \quad u \quad \frac{1}{u^2} \quad \frac{\cos u}{\sin u^3}$	
β	10	

Supplementary Table 13. The detailed information of Phototaxis Robot data.

Original data points	2000
Sampling rate	1
Used data points	2000
Number of points (subsystem 1)	654
Number of points (subsystem 2)	585
Number of points (subsystem 3)	761

Supplementary Table 14. The transition distribution for Phototaxis Robot example.

System	Subsystem 1	Subsystem 2	Subsystem 3
Subsystem 1		14	15
Subsystem 2	10		14
Subsystem 3	18	11	

Supplementary Table 15. The identified result and details of tuning parameters in the Phototaxis Robot example without reduntant dictionary functions.

Noise	$N_p = 0$	$N_p = 2\%$	$N_p = 4\%$	$N_p = 6\%$
Library$\mathbf{\Phi}$		$u_1 - u_2 \qquad \frac{1}{u_1 - u_2}$		
Identified subsystem 1	$y = -0.9980$ $(u_1 - u_2)$	$y = -0.9978$ $(u_1 - u_2)$	$y = -0.9964$ $(u_1 - u_2)$	$y = -0.9955$ $(u_1 - u_2)$
Identified subsystem 2	$y = \frac{0.9908}{u_1 - u_2}$	$y = \frac{0.9947}{u_1 - u_2}$	$y = \frac{0.9820}{u_1 - u_2}$	$y = \frac{0.9821}{u_1 - u_2}$
Identified subsystem 3	$y = 0$	$y = 0$	$y = 0.0068(u_1 - u_2)$	$y = 0.0095(u_1 - u_2)$
$\lambda_\mathbf{z}$	$5e-4$	$5e-4$	$5e-4$	0.001
$\epsilon_\mathbf{z}$		$1e-4$		
$\lambda_\mathbf{w}$	0.05	0.05	0.1	0.1
$\epsilon_\mathbf{w}$	0.05	0.06	0.2	0.2
Number of misclassified points	0	11	28	47

Supplementary Table 16. The identified result and details of tuning parameters in the Phototaxis Robot example with redundant dictionary functions.

Noise	$N_p = 0$	$N_p = 2\%$	$N_p = 4\%$	$N_p = 6\%$
Library $\mathbf{\Phi}$		$1 \quad u_1 - u_2 \quad \frac{1}{u_1 - u_2} \quad u_1^2 \quad u_2^2$		
Identified subsystem 1	$y = -1.0000$ $(u_1 - u_2)$	$y = -0.9957$ $(u_1 - u_2)$	$y = -0.9944$ $(u_1 - u_2)$	$y = -0.9941$ $(u_1 - u_2)$
Identified subsystem 2	$y = \frac{0.9998}{u_1 - u_2}$	$y = \frac{0.9891}{u_1 - u_2}$	$y = \frac{0.9727}{u_1 - u_2}$	$y = \frac{0.9689}{u_1 - u_2}$
Identified subsystem 3	$y = 0$	$y = -0.0002u_2^2$	$y = 0.0046(u_1 - u_2) + 0.0014u_1^2$	$y = 0.0064(u_1 - u_2) + 0.0019u_1^2$
$\lambda_{\mathbf{z}}$	$1e-4$	$1e-4$	$5e-4$	$1e-3$
$\epsilon_{\mathbf{z}}$		$1e-4$		
$\lambda_{\mathbf{w}}$	$1e-3$	0.1	0.15	0.15
$\epsilon_{\mathbf{w}}$	0.005	0.06	0.2	0.2
Number of misclassified points	0	11	28	48

Supplementary Table 17. The identified result and details of tuning parameters in the Phototaxis Robot example.

System	Subsystem 1	Subsystem 2	Subsystem 3
Subsystem 1		$u_3 < 0.4986u_4$	$u_3 < 0.5085u_5$
Subsystem 2	$u_4 < 0.4553u_3$		$u_4 < 0.5055u_5$
Subsystem 3	$u_5 < 0.5242u_3$	$u_5 < 0.4543u_4$	
Library $\mathbf{\Psi}$		$1 \quad u_3 \quad u_4 \quad u_5$	
β		0.5	

Supplementary Table 18. The identified transition logic for the Phototaxis Robot example.

System	Subsystem 1	Subsystem 2	Subsystem 3					
Subsystem 1		$u_3 < 0.4986u_4$	$u_3 < 0.5085u_5$					
Subsystem 2	$u_4 < 0.4553u_3$		$u_4 < 0.5055u_5$					
Subsystem 3	$u_5 < 0.5242u_3$	$u_5 < 0.4543u_4$						
Library $\boldsymbol{\Psi}$	1	u_1^{-1}	u_2	$\sin(u_1)$	$\cos(u_2)$	$e^{u_1 u_2}$	u_3	u_4	u_5
β	0.5							

Supplementary Table 19. The detailed information of Non-linear Hybrid System data.

Original data points	2000
Sampling rate	1
Used data points	2000
Number of points (subsystem 1)	605
Number of points (subsystem 2)	738
Number of points (subsystem 3)	657

Supplementary Table 20. The transition distribution for Non-linear Hybrid System example.

System	Subsystem 1	Subsystem 2	Subsystem 3
Subsystem 1			157
Subsystem 2	158		
Subsystem 3		157	

Supplementary Table 21. The identified result and details of Non-linear Hybrid System.

Noise	$N_p = 0$	$N_p = 2\%$	$N_p = 4\%$	$N_p = 6\%$
Library$\boldsymbol{\Phi}$	$u_1 u_2 \qquad \frac{6u_1}{6+u_2} \qquad \frac{u_1+u_2}{u_1-u_2}$			
Identified subsystem 1	$y = 0.9998u_1u_2$	$y = 0.9958u_1u_2$	$y = 0.9962u_1u_2$	$y = 0.9953u_1u_2$
Identified subsystem 2	$y = 0.9983\frac{6u_1}{6+u_2}$	$y = 0.9961\frac{6u_1}{6+u_2}$	$y = 0.9947\frac{6u_1}{6+u_2}$	$y = 0.9939\frac{6u_1}{6+u_2}$
Identified subsystem 3	$y = 0.9990\frac{u_1+u_2}{u_1-u_2}$	$y = 0.9945\frac{u_1+u_2}{u_1-u_2}$	$y = 0.9901\frac{u_1+u_2}{u_1-u_2}$	$y = 0.9949\frac{u_1+u_2}{u_1-u_2}$
$\lambda_{\mathbf{z}}$	$1e-6$	$1.5e-4$	$1.5e-4$	$1.5e-4$
$\epsilon_{\mathbf{z}}$	$1e-4$			
$\lambda_{\mathbf{w}}$	0.03			
$\epsilon_{\mathbf{w}}$	0.6	2	2	2
Number of misclassified points	0	63	129	177

Supplementary Table 22. The identified result and details of Non-linear Hybrid Systems when there exists redundant dictionary functions.

Noise	$N_p = 0$	$N_p = 2\%$	$N_p = 4\%$	$N_p = 6\%$
Library $\boldsymbol{\Phi}$	$u_1 u_2 \quad \frac{6u_1}{6+u_2}$	$\frac{u_1+u_2}{u_1-u_2} \quad u_1 \quad u_2$	$\sin(u_1) \quad \sin(u_2)$	$u_1^2 \quad u_2^2$
Identified subsystem 1	$y = 0.9997u_1u_2$	$y = 0.9999u_1u_2$	$y = 0.9944u_1u_2$	$y = 0.9953u_1u_2$
Identified subsystem 2	$y = 0.9983\frac{6u_1}{6+u_2}$	$y = 0.9960\frac{6u_1}{6+u_2}$	$y = 0.9960\frac{6u_1}{6+u_2}$	$y = 0.9963\frac{6u_1}{6+u_2}$
Identified subsystem 3	$y = 0.9990\frac{u_1+u_2}{u_1-u_2}$	$y = 0.9975\frac{u_1+u_2}{u_1-u_2}$	$y = 0.9898\frac{u_1+u_2}{u_1-u_2}$	$y = 0.9877\frac{u_1+u_2}{u_1-u_2}$
$\lambda_{\mathbf{z}}$	$1e-5$	$5e-5$	$1.5e-4$	$1.5e-4$
$\epsilon_{\mathbf{z}}$	$1e-4$			
$\lambda_{\mathbf{w}}$	0.03	0.03	0.032	0.0282
$\epsilon_{\mathbf{w}}$	0.6	0.8	2	2
Number of misclassified points	0	67	129	175

Supplementary Table 23. The identified transition logic of Non-linear Hybrid System.

System	Subsystem 1	Subsystem 2	Subsystem 3
Subsystem 1			$u_1{}^2 + u_2{}^2 < 8.9993$
Subsystem 2	$u_1{}^2 + u_2{}^2 > 24.9803$		
Subsystem 3		$u_1 u_2 < -0.013$	
Library $\boldsymbol{\Psi}$	$1 \quad u_1 u_2 \quad u_1{}^2 + u_2{}^2$		
β	0.01		

Supplementary Table 24. The identified transition logic of Non-linear Hybrid System when there are redundant dictionary functions.

System	Subsystem 1	Subsystem 2	Subsystem 3
Subsystem 1			$30.8698u_1{}^2 + 31.9412u_2{}^2$ < 282.0993
Subsystem 2	$11.7323u_1{}^2 + 11.7384u_2{}^2$ > 292.7959		
Subsystem 3		$u_1u_2 < -0.013$	
Library $\mathbf{\Psi}$	$1 \quad u_1 \quad u_2 \quad e^{u_1+u_2} \quad u_1u_2 \quad u_1{}^2 \quad u_2{}^2$		
β	0.01		

Supplementary Table 25. The detailed information of experiment data from car.

Original data points	400
Used data points	396
Number of points (Straightway)	147
Number of points (curve)	249
Number of transitions (straightway to curve)	2
Number of transitions (curve to straightway)	2

Supplementary Table 26. The identified transition logic of autonomous car testbed.

System	Straightaway	Curve
Straightaway		$straight < 0.3318$
Curve	$straight > 0.6072$	
Library $\mathbf{\Psi}$	$1 \quad straight \quad \sin(v(k)) \quad \cos(v(k)) \quad \tan(v(k)) \quad \frac{v(k-1)-v(k-4)}{v(k-2)} \quad v(k-1)\tan(v(k-3))$	
β	0.001	

Supplementary Table 27. The identified result and details of autonomous car testbed.

Library $\mathbf{\Phi}$	$1 \quad v(k) \quad v(k-1)$	
Speed control strategy	Straightaway	Curve
True strategy	$\Delta u(k) = 9.5(380 - v(k))$ $+48(v(k-1) - v(k))$	$\Delta u(k) = 9.5(280 - v(k))$ $+48(v(k-1) - v(k))$
True times	147	249
Identified strategy	$\Delta u(k) = 9.4957(379.9550 - v(k))$ $+47.9742(v(k-1) - v(k))$	$\Delta u(k) = 9.4960(279.9462 - v(k))$ $+47.9888(v(k-1) - v(k))$
Identified switching times	147	249
$\lambda_{\mathbf{z}}$	0.01	
$\epsilon_{\mathbf{z}}$	100	
$\lambda_{\mathbf{w}}$	$1e-5$	
$\epsilon_{\mathbf{w}}$	8	

Supplementary Table 28. The identified result and details of autonomous car testbed with redundant dictionary functions.

Library $\mathbf{\Phi}$	all the polynomial combinations of $v(k),\ldots,v(k-4)$ to fourth order	
Speed control strategy	Straightaway	Curve
True strategy	$\Delta u(k) = 9.5(380 - v(k))$ $+48(v(k-1) - v(k))$	$\Delta u(k) = 9.5(280 - v(k))$ $+48(v(k-1) - v(k))$
True times	147	249
Identified strategy	$\Delta u(k) = 9.4957(379.9554 - v(k))$ $+47.9741v(k-1) - v(k))$	$\Delta u(k) = 9.4959(279.9465 - v(k))$ $+47.9888(v(k-1) - v(k))$
Identified switching times	147	249
$\lambda_{\mathbf{z}}$	0.01	
$\epsilon_{\mathbf{z}}$	100	
$\lambda_{\mathbf{w}}$	$1e-5$	
$\epsilon_{\mathbf{w}}$	8	

Supplementary Table 29. The detailed information of experiment data from Chua's circuit.

Original data points	120000
Sampling rate	$5 \times 10^6 \text{Hz}$
Down sampling	1:50:120000
Used data points	177
Number of points (subsystem 1)	57
Number of points (subsystem 2)	21
Number of points (subsystem 3)	99

Supplementary Table 30. The transition distribution of experiment data from Chua's circuit.

System	Subsystem 1	Subsystem 2	Subsystem 3
Subsystem 1		2	
Subsystem 2	2		1
Subsystem 3		2	

Supplementary Table 31. True parameters of the built Chua's circuit.

Item	Value	Item	Value
a	$-1.2309e-3$	c_1	$0.01\mu F$
b	$-8.743e-4$	c_2	$0.1\mu F$
b'	$-8.864e-4$	dt	$10^{-5}s$
threshold	1.5	L	$6.8mH$
R	921		

Supplementary Table 32. The identified subsystems and transition logic of Chua's circuit which contains all subsystems with redundant dictionary functions.

Library $\mathbf{\Phi}$	$1 \; y_1 \; y_2 \; e^{y_1} \; \frac{y_1}{y_2} \; \frac{\cos(0.1y_1)^2}{1+y_2^2} \; \cos(y_1 + y_2)^2$		
Identified subsystems	$10^{-5}\frac{dy_1}{dt} = 1.0758y_2 - 0.2028y_1 - 0.5405$		$y_1 < -1.4348$
	$10^{-5}\frac{dy_1}{dt} = 1.0793y_2 + 0.1576y_1$		$-1.5627 < y_1 < 1.3137$
	$10^{-5}\frac{dy_1}{dt} = 1.0869y_2 - 0.2127y_1 + 0.4859$		$y_1 > 1.4627$
$\lambda_{\mathbf{z}}$	0.05		
$\epsilon_{\mathbf{z}}$	0.012		
$\lambda_{\mathbf{w}}$	0.01		
$\epsilon_{\mathbf{w}}$	0.044		
Library $\mathbf{\Psi}$	$1 \quad y_1 \quad y_2 \quad \sin(y_2)\cos(y_1) \quad \frac{\frac{dy_1}{dt}}{\sin(y_1)+\frac{dy_1}{dt}} \quad \frac{\frac{dy_1}{dt}}{y_2} \quad \frac{dy_1}{dt}$		
β	0.01		

Supplementary Table 33. The detailed information of experiment data from wind turbine system platform.

Original data points	20701
Sampling rate	1000Hz
Down sampling	1:300:20701
Used data points	64
Number of points (normal)	61
Number of points (fault)	3
Number of transitions	1

Supplementary Table 34. The identified result and details of the gearbox broken tooth fault with redundant dictionary functions.

System	Gearbox
True fault time	68
Identified fault time	68
$\lambda_{\mathbf{z}}$	1.5
$\epsilon_{\mathbf{z}}$	$1e-4$
$\lambda_{\mathbf{w}}$	$5e-5$
$\epsilon_{\mathbf{w}}$	0.026
Library $\mathbf{\Phi}$	all the polynomial combinations of $y(k) \cdots y(k-5)$ to second order
Number of misclassified points	0
Library $\mathbf{\Psi}$	$1\ k$
β	0.1

Supplementary Table 35. The detailed information of data sampled during power system line fault condition.

Original data points	40
Used data points	40
Number of points (normal)	30
Number of points (fault)	10
Normal to fault	1

Supplementary Table 36. The identified result and details of power system fault detection.

Bus	Bus 6 and bus 12	Other bus except bus 1	Bus 1
True time for fault occurance	31	None	None
Identified time for fault occurance	31	None	None
$\lambda_{\mathbf{z}}$	$1e-3$	$1e-3$	$1e-3$
$\epsilon_{\mathbf{z}}$	0.008	0.008	0.008
$\lambda_{\mathbf{w}}$	$1e-6$	$1e-6$	$1e-9$
$\epsilon_{\mathbf{w}}$	0.05	0.05	0.05
Library $\boldsymbol{\Psi}$	$1\ t$		
β	0.01	None	None

Supplementary Table 37. The detailed information of smart gird data.

Original data points	180
Used data points	180
Number of points (subsystem 1)	90
Number of points (subsystem 2)	90
Number of transitions (S1 to S2)	3
Number of transitions (S2 to S1)	2

Supplementary Table 38. The detailed parameters of switch operators.

Transition rules	Time	Opened switch	Closed switch	Bus of load change
$\mathcal{T}_{1\to2}$	$31, 91, 151$	$11 - 12$	$12 - 22$	$9, 10, 11$
$\mathcal{T}_{2\to1}$	$61, 121$	$12 - 22$	$11 - 12$	$20, 21, 22$

Supplementary Table 39. The identified result and detailed parameters.

System model	Subsystem 1	Subsystem 2
True switching time	$31, 91, 151$	$61, 121$
Identified switching time	$31, 91, 151$	$61, 121$
$\lambda_{\mathbf{z}}$	$5e - 3$	
$\epsilon_{\mathbf{z}}$	$1.5e - 2$	
$\lambda_{\mathbf{w}}$	$1e - 6$	
$\epsilon_{\mathbf{w}}$	$5e - 2$	
Number of misclassified points	0	

Supplementary Table 40. The identified transition logic for the model switching in smart grid.

System	Subsystem 1	Subsystem 2
Subsystem 1		$\Delta U_{10} < -0.0498$
Subsystem 2	$\Delta U_{21} < -0.0471$	
Library $\mathbf{\Psi}$	$1 \quad \Delta U_1 \cdots \Delta U_{33}$	
β	$4.5e - 5$	

Supplementary Table 41. The detailed information of data sampled from AP model.

Original data points	120001
Sampling rate	200Hz
Down sampling	24000:60:100000
Used data points	1267
Number of points (subsystem 1)	707
Number of points (subsystem 2)	560
Number of transitions (S1 to S2)	1

Supplementary Table 42. The identified transition logic for gating variable h.

gating variable h	Subsystem 1	Subsystem 2
Subsystem 1		$V \leq -40.0093$
Library $\boldsymbol{\Psi}$	1	V
β	1e-6	

Supplementary Table 43. The identified results and detailed parameters of AP model.

Gating variable	h	j
Actual subsystem 1	$\dot{h} = -h\beta_{h2}$	$\dot{j} = -0.3j\frac{\exp(-2.535\times10^{-7}V)}{1+\exp[-0.1(V+32)]}$
Actual subsystem 2	$\dot{h} = 0.135\exp(-\frac{V+80}{6.8}) - 0.135h\exp(-\frac{V+80}{6.8}) - h\beta_{h1}$	$\dot{j} = \alpha_{j1} - j\alpha_{j1} - 0.1212j\frac{\exp(-0.01052V)}{1+\exp[-0.1378(V+40.14)]}$
Actual change time	332.10 ms	332.10 ms
Identified subsystem 1	$\dot{h} = -0.9999h\beta_{h2}$	$\dot{j} = -0.3000j\frac{\exp(-2.535\times10^{-7}V)}{1+\exp[-0.1(V+32)]}$
Identified subsystem 2	$\dot{h} = 0.1349\exp(-\frac{V+80}{6.8}) - 0.1349\exp(-\frac{V+80}{6.8})h - 0.9987h\beta_{h1}$	$\dot{j} = 1.0000\alpha_{j1} - 1.0000j\alpha_{j1} - 0.1212j\frac{\exp(-0.01052V)}{1+\exp[-0.1378(V+40.14)]}$
Identified change time	332.10 ms	332.10 ms
$\lambda_{\mathbf{z}}$	1e-4	1e-4
$\epsilon_{\mathbf{z}}$	3e-5	3e-5
$\lambda_{\mathbf{w}}$	3e-5	1e-5
$\epsilon_{\mathbf{w}}$	5e-5	5e-5
Library $\mathbf{\Phi}$	$\exp(-\frac{V+80}{6.8})$ $h\exp(-\frac{V+80}{6.8})$ $h\beta_{h1}$ $h\beta_{h2}$	α_{j1} $j\alpha_{j1}$ $j\frac{\exp(-0.01052V)}{1+\exp[-0.1378(V+40.14)]}$ $j\frac{\exp(-2.535\times10^{-7}V)}{1+\exp[-0.1(V+32)]}$
Number of misclassified points	0	0

Supplementary Table 44. The detailed parameters and identified results of AP model with polynomial dictionary functions.

Gating variable	h	j
Actual subsystem 1	$\dot{h} = -h\beta_{h2}$	$\dot{j} = -0.3j\frac{\exp(-2.535\times10^{-7}V)}{1+\exp[-0.1(V+32)]}$
Actual subsystem 2	$\dot{h} = 0.135\exp(-\frac{V+80}{6.8}) - 0.135h\exp(-\frac{V+80}{6.8}) - h\beta_{h1}$	$\dot{j} = \alpha_{j1} - j\alpha_{j1} - 0.1212j\frac{\exp(-0.01052V)}{1+\exp[-0.1378(V+40.14)]}$
Actual change time	332.10 ms	332.10 ms
Identified subsystem 1	$\dot{h} = 0.0006 - 0.0003V - 2.1791h - 0.0608hV - 0.2323V^2 - 0.0004hV^2 - 0.0032h^2V - 0.0256V^3$	$\dot{j} = 0.0610j - 0.0218j^2$
Identified subsystem 2	$\dot{h} = -1.0506h^2 + 0.0762hV - 5.9332\times 10^{32}h^3 - 0.0033hV^2 + 3.9061 \times 10^{31}h^2V + 7.9368 \times 10^{64}h^3$	$\dot{j} = -0.2525j + 0.0003j^3$
Identified change time	331.80 ms	331.80 ms
$\lambda_{\mathbf{z}}$	1e-4	1e-5
$\epsilon_{\mathbf{z}}$	3e-5	3e-5
$\lambda_{\mathbf{w}}$	3e-8	1e-4
$\epsilon_{\mathbf{w}}$	5e-5	5e-5
Library $\mathbf{\Phi}$	polynomials of V, h up to third order	polynomials of V, j up to third order
Number of misclassified points	1	1

Supplementary Table 45. The identified transition logic for gating variable j.

gating variable j	Subsystem 1	Subsystem 2
Subsystem 1		$V \leq -40.0093$
Library $\boldsymbol{\Psi}$		1 V
β		1e-6

Supplementary Table 46. **The details of simulation datasets in [22].**

Systems	Form	Data used to train [22]		Data used to train IHYDE	
		time	points	time	points
Linear 2D	$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.1 & 2 \\ -2 & -0.1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$	$t \in [0, 25]$	2501	$t \in [0, 10]$	1001
Cubic 2D	$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.1 & 2 \\ -2 & -0.1 \end{bmatrix}\begin{bmatrix} x^3 \\ y^3 \end{bmatrix}$	$t \in [0, 25]$	2501	$t \in [0, 10]$	1001
Linear 3D	$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -0.1 & 2 & 0 \\ -2 & -0.1 & 0 \\ 0 & 0 & -0.3 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}$	$t \in [0, 50]$	5001	$t \in [0, 10]$	1001
Logistic map	$x_{k+1} = \mu_k x_k (1 - x_k)$ $\mu_{k+1} = \mu_k$		9990		990
Lorenz system	$\dot{x} = 10y - 10x$ $\dot{y} = 28x - xz - y$ $\dot{z} = xy - 2.6667z$	$t \in [0.001, 100]$	100000	$t \in [0.01, 10]$	1000
Lorenz TVDiff	$\dot{x} = 10y - 10x$ $\dot{y} = 28x - xz - y$ $\dot{z} = xy - 2.6667z$	$t \in [0.001, 50]$	48002	$t \in [0.001, 50]$ downsampling=25	1921
Hopf TVDiff	$\dot{x} = ux - y - x^3 - xy^2$ $\dot{y} = x + uy - yx^2 - y^3$ $\dot{u} = 0$		399014	downsampling=100	3991

Supplementary Table 47. **The identified results using datasets in [22].** Seven prototypical systems were examined, IHYDE successfully discovers *all* of them.

	Identified systems by [22]	Identified systems by IHYDE
Linear 2D	$\dfrac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.1015 & 2.0027 \\ -1.9990 & -0.0994 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$	$\dfrac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.0993 & 2.0054 \\ -2.0004 & -0.1048 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$
Cubic 2D	$\dfrac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.0996 & 1.9970 \\ -1.9994 & -0.0979 \end{bmatrix}\begin{bmatrix} x^3 \\ y^3 \end{bmatrix}$	$\dfrac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.1015 & 2.0005 \\ -2.0010 & -0.1002 \end{bmatrix}\begin{bmatrix} x^3 \\ y^3 \end{bmatrix}$
Linear 3D	$\dfrac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -0.0996 & 2.0005 & 0 \\ -1.9997 & -0.0994 & 0 \\ 0 & 0 & -0.3003 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}$	$\dfrac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -0.0992 & 2.0002 & 0 \\ -1.9999 & -0.0991 & 0 \\ 0 & 0 & -0.2983 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}$
Logistic map	$x_{k+1} = \mu_k x_k(0.9993 - 0.9989 x_k)$ $\mu_{k+1} = 1.0000\mu_k$	$x_{k+1} = \mu_k x_k(1.0005 - 1.0006 x_k)$ $\mu_{k+1} = 1.0000\mu_k$
Lorenz system	$\dot{x} = 9.9998y - 9.9996x$ $\dot{y} = 27.9980x - 0.9999xz - 0.9997y$ $\dot{z} = 1.0000xy - 2.6665z$	$\dot{x} = 10.0060y - 9.9968x$ $\dot{y} = 27.9480x - 0.9957xz - 0.9954y$ $\dot{z} = 1.0010xy - 2.6673z$
Lorenz TVDiff	$\dot{x} = 9.9999y - 9.9856x$ $\dot{y} = 27.7382x - 0.9949xz - 0.8763y$ $\dot{z} = 1.0000xy - 2.6618z$	$\dot{x} = 10.0087y - 10.0227x$ $\dot{y} = 27.6620x - 0.9934xz - 0.8461y$ $\dot{z} = 0.9993xy - 2.6640z$
Hopf TVDiff	$\dot{x} = 0.9269ux - 0.9920y - 0.9208x^3$ $-0.9211xy^2$ $\dot{y} = 0.9914x + 0.9294uy - 0.9244yx^2$ $-0.9252y^3$ $\dot{u} = 0$	$\dot{x} = 0.9193ux - 0.9921y - 0.9109x^3$ $-0.9179xy^2$ $\dot{y} = 0.9911x + 0.9164uy - 0.9127yx^2$ $-0.9130y^3$ $\dot{u} = 0$

Supplementary Table 48. The tuning parameters are presented for these prototypical examples.

	Noise	$\lambda_{\mathbf{z}}$	$\epsilon_{\mathbf{z}}$	$\lambda_{\mathbf{w}}$	$\epsilon_{\mathbf{w}}$
Linear 2D	0.05	1	$1e-4$	$2e-3$	0.2
Cubic 2D	0.05	1	$1e-4$	$2e-3$	0.2
Linear 3D	0.01	1	$1e-4$	$2e-3$	0.05
Logistic map	0.01	1	$1e-4$	$2e-3$	0.05
Lorenz system	1	1	$1e-4$	$1e-3$	4
Lorenz TVDiff	0.01	1	$1e-4$	$4e-3$	3
Hopf TVDiff	0.005	1	$1e-4$	$3e-3$	0.1

Supplementary Table 49. The selected hyperparameters and the identified subsystems for example 3.

Library $\mathbf{\Phi}$	polynomials of x up to fifth order
Actual subsystem 1	$\dot{x} = -x^3$
Identified subsystem 1	$\dot{x} = -0.9975x^3$
Actual subsystem 2	$\dot{x} = -\cos(x)$
Identified subsystem 2	$\dot{x} = -0.9960 + 0.4651x^2$
$\lambda_{\mathbf{z}}$	$1e-6$
$\epsilon_{\mathbf{z}}$	$1e-2$
$\lambda_{\mathbf{w}}$	$1e-4$
$\epsilon_{\mathbf{w}}$	0.1826

Supplementary Table 50. Directories in the IHYDE toolbox.

Directories	Description
/IHYDE	main functions and examples
/IHYDE/docs	pdf and html documentation
/IHYDE/data	datasets we used in the paper
/IHYDE/tools	functions for IHYDE

Supplementary Table 51. The introduction of function library which can construct library for further identification.

Function library	Description
yin	an m by n matrix which contains time-course input-output data. In here, m is the sample number, and n is the number of variables.
polyorder	used to construct the polynomial of the highest order (up to fifth order).
basis_function	add more dictionary functions. It can be turned off, if basis_function.work set as 'off'.
Phi	constructed dictionary matrix Φ.

Supplementary Table 52. The introduction of function ihyde. The ihyde can be used to identify each subsystem.

Function ihyde	Description
parameter.y	the output data.
parameter.normalize_y	set to 1 if y need to be normalized.
parameter.max_s	the max number of subsystems that could be identified by IHYDE.
parameter.epsilon	a 1 by 2 vector. For finding new subsystems.
parameter.lamdba	a 1 by 2 vector. For the identification of a new subsystems.
parameter.MAXITER	the max number of iterations that the sparsesolver function solves.
result. idx_sys	the index of each subsystem.
result.sys	the model of each subsystem.
result.theta	z of each identified subsystem.
result.error	the fitting error.

Supplementary Table 53. The introduction of function finetuning. Based on the minimum error principle, it finetunes the result from ihyde and outputs the final result.

Function finetuning	Description
result.lambda	the trading-off parameter λ of the `sparsesolver` function. Parameter.lamdba(2) is set as the default value.
result.epsilon	the threshold in finetuning. Parameter.epsilon(2) is set as the default value.
result.threshold	the threshold for subsystem clustering.
final_result.idx	the index of each subsystem.
final_result.sys	the model of each subsystem.
final_result.allerror	the error which compared with the true output.

Supplementary Table 54. The introduction of function ihydelogic.

Function ihydelogic	Description
para_log.Phi2	constructed dictionary matrix Ψ for inferring transition logic of each subsystem.
para_log.idx_sys	the index of each subsystem.
para_log.beta	the tradeoff parameter in the ℓ_1 sparse logistic regression.
para_log.y	the output data.
para_log.normalize	set to 1 if Phi2 need to be normalized.

SUPPLEMENTARY NOTES

Supplementary Note 1: Notations

$\mathbb{R}^n$: denotes the n-dimensional Euclidean space.

$\mathbb{Z}$: denotes the set of integers, $\ldots, -1, 0, 1, \ldots$.

$\|\mathbf{x}\|_{\ell_0}$: the ℓ_0-norm of a vector $\mathbf{x}$, i.e., $\|\mathbf{x}\|_{\ell_0} = \sum_{i=1}^{n} |x_i|^0$ (defining $0^0 = 0$).

$\|\mathbf{x}\|_{\ell_1}$: the ℓ_1-norm of a vector $\mathbf{x}$, i.e., $\|\mathbf{x}\|_{\ell_1} = \sum_{i=1}^{n} |x_i|$.

$\|\mathbf{x}\|_{\ell_2}$: the ℓ_2-norm of a vector $\mathbf{x}$, i.e., $\|\mathbf{x}\|_{\ell_2} = (\sum_{i=1}^{n} |x_i|^2)^{1/2}$.

$\|\mathbf{A}\|_F$: the Frobenius-norm of a matrix $\mathbf{A}$, i.e., $\|\mathbf{A}\|_F = (\text{trace}(\mathbf{A}^T \mathbf{A}))^{1/2}$.

$\mathbf{A}$: For a matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$, $\mathbf{A}[i, j] \in \mathbb{R}$ denotes the element in the ith row and j^{th} column, $\mathbf{A}[i, :] \in \mathbb{R}^{1 \times N}$ denotes its ith row, $\mathbf{A}[:, j] \in \mathbb{R}^{M \times 1}$ denotes its jth column.

$\boldsymbol{\alpha}$: For a column vector $\boldsymbol{\alpha} \in \mathbb{R}^{N \times 1}$, $\boldsymbol{\alpha}[i]$ denotes its ith element.

$\mathbf{I}_k$: A k-dimensional identity matrix.

$\mathbf{0}_k$: A k-dimensional zero matrix.

Supplementary Note 2: Introduction to Hybrid Dynamical Systems

A dynamical system describes how state variables (typically physical quantities) evolve with respect to time. Following definitions in [20], we define three types of variables.

1. continuous state variables: if the state variable takes value in $\mathbb{R}^n$ for $n \geq 1$.

2. discrete state variables: if the state variable takes value in a finite set, for example, $\{1, 2, 3, \ldots\}$.

3. hybrid state variables: if a part of the state variables are continuous and the other discrete.

Based on the time set over which the state evolves, we classify the dynamical systems as:

- continuous time: if the set of time is a subset of the real line $\mathbb{R}$. Normally we use $t \in \mathbb{R}$ to denote the continuous time. The evolution of the state-variables in continuous time can be described as ordinary differential equations.

- discrete time: if the set of time is a subset of the integers. Normally we use $k \in \mathbb{Z}$ to denote discrete time. The evolution of the state-variables in discrete time can be described as difference equations.

A hybrid dynamical system $\mathcal{H}$, is defined as a tuple, $\mathcal{H} = (\mathcal{W}, \mathcal{M}, \mathcal{F}, \mathcal{T})$ with the following definitions:

- $\mathcal{W}$ defines a subspace in $\mathbb{R}^{m+n}$ for input-output variables $\mathbf{u}(t) \in \mathbb{R}^m, \mathbf{y}(t) \in \mathbb{R}^n$;

- $\mathcal{M}$ defines a countable, discrete set of modes in which only a single mode, $m(t) \in \{1, 2, \ldots, K\}$, is occupied at a given time;

- $\mathcal{F}$ defines a countable discrete set of first-order differential equations:

$$\mathcal{F} = \left\{ \frac{d\mathbf{y}(t)}{dt} = \mathbf{F}_k\left(\mathbf{y}(t), \mathbf{u}(t)\right)) \mid k = 1, 2, \ldots, K \right\}$$

- $\mathcal{T}$ defines a countable discrete set of transitions, where $\mathcal{T}_{i \to j}$ denotes a Boolean expression that represents the condition to transfer from mode i to j.

The signals $\mathbf{y}(t)$ and $\mathbf{u}(t)$ are sampled at a rate $h > 0$, i.e. sampled at times $0, h, 2h, 3h....$ For fast enough sampling (or low h), standard system identification typically obtains first a discrete-time system, and then coverts it to a continuous-time system [21]. One of the simplest methods to approximate derivatives is to consider

$$\frac{d\mathbf{y}(t)}{dt} \approx \frac{\mathbf{y}(t+h) - \mathbf{y}(t)}{h}$$

which yields the discrete-time system

$$\mathbf{y}(t+h) = \mathbf{y}(t) + h\, \mathbf{F}_k(\mathbf{y}(t), \mathbf{u}(t)) \triangleq \mathbf{f}_k(\mathbf{y}(t), \mathbf{u}(t)), \qquad k \in \{1, 2, \ldots, K\}.$$

For simplification of notation, assume the system can be written as

$$\mathbf{y}(t+h) = \mathbf{f}_k(\mathbf{y}(t), \mathbf{u}(t)) \triangleq \mathbf{I}_k(\mathbf{y}(t)) + \mathbf{h}_k(\mathbf{u}(t)), \qquad k \in \{1, 2, \ldots, K\}.$$

Hence, the class of systems considered is discrete-time, Markovian and nonlinear. While this is already a very rich class of systems, it can be easily extended to more general nonlinear systems, including, for example, dynamics of non-separable nonlinear functions of $(\mathbf{y}(t), \mathbf{u}(t))$.

Without loss of generality, we can rescale the time variable t so that $h = 1$. Thus, we can construct a mathematical model for hybrid dynamical systems

$$m(t+1) = \mathcal{T}(m(t), \mathbf{y}(t), \mathbf{u}(t))$$

$$\mathbf{y}(t+1) = \mathbf{f}(m(t), \mathbf{y}(t), \mathbf{u}(t)) = \begin{cases} \mathbf{f}_1(\mathbf{y}(t), \mathbf{u}(t)), & \text{if } m(t) = 1, \\ \quad\quad\vdots, & \quad\vdots \\ \mathbf{f}_K(\mathbf{y}(t), \mathbf{u}(t)), & \text{if } m(t) = K. \end{cases} \quad (1)$$

Example 1 *Consider again the temperature control system in Supplementary Figure 1, consisting of a heater and a thermostat. The variables in this model are the room temperature $y(t) \in \mathbb{R}$ and the operating mode of the heater (on or off). Assuming a sampling time of $h > 0$, we obtain the following approximate difference equations (discretized from an ordinary differential equation) for the temperature*

Subsystem 1 (heat off) : $\dfrac{y(t+1) - y(t)}{h} \approx -ay(t), \Rightarrow y(t+1) = (1 - ah)y(t).$

Subsystem 2 (heat on) : $\dfrac{y(t+1) - y(t)}{h} \approx -a(y(t) - 30), \Rightarrow y(t+1) = (1 - ah)y(t) + 30ah.$

These two equations model how the temperature changes under the heater off or on, respectively. The transition logics between the two subsystems are

Transition logic from subsystem 1 to 2 $\mathcal{T}_{1 \to 2}$: $y \leq 19$

Transition logic from subsystem 2 to 1 $\mathcal{T}_{2 \to 1}$: $y \geq 21$,

representing the controller of the operating mode of the heater. Given this hybrid dynamical system, we can study its stability or simulate it to check possible state trajectories. Note that, in practice, hybrid dynamical systems, such as this one, are usually unknown or only partially known. The goal of this paper is to infer both the above subsystems and the transition logic (Fig. 1(a)) from only time-series data of the temperature in Fig. 1(c).

SUPPLEMENTARY METHODS

Supplementary Method 1: IHYDE Algorithm

When a hybrid dynamical system has a single subsystem, i.e., $K = 1$ in Eq. (1), it becomes a time-invariant nonlinear dynamical system. We start by briefly reviewing identification

47

tools for this class of systems from [10, 22], since parts of our proposed algorithm are based on these tools. As explained before, our algorithm uses only time-series data to directly model the system. Hence, the first step is to collect time-course input-output data $(\mathbf{y}(t), \mathbf{u}(t))$ uniformly sampled at a number of discrete time indices $t = 1, 2, \ldots, M + 1$. Let

$$
\mathbf{Y} = \begin{bmatrix} | & | & | & | \\ \mathbf{y}(1) & \mathbf{y}(2) & \cdots & \mathbf{y}(M) \\ | & | & | & | \end{bmatrix}^{T}, \quad \mathbf{U} = \begin{bmatrix} | & | & | & | \\ \mathbf{u}(1) & \mathbf{u}(2) & \cdots & \mathbf{u}(M) \\ | & | & | & | \end{bmatrix}^{T}.
$$

Note that $\mathbf{y}(t) \in \mathbb{R}^n$ and $\mathbf{u}(t) \in \mathbb{R}^m$, and so $\mathbf{Y} \in \mathbb{R}^{M \times n}$ and $\mathbf{U} \in \mathbb{R}^{M \times m}$. Next, we construct an overdetermined library $\mathbf{\Phi}(\mathbf{Y}, \mathbf{U})$ consisting of potential nonlinear functions that appear in $\mathbf{f}_k$ in Eq. (1). It is expected that the true nonlinearities are part of this library in order to recover the true dynamics. The choice of these functions is guided by the particular field of study. For example, the library would consist of sinusoidal functions in pendulums, and polynomial and sigmoidal functions in biochemical networks. As an illustration, a library consisting of constant or polynomials would result in the following dictionary matrix

$$
\mathbf{\Phi}(\mathbf{Y}, \mathbf{U}) = \begin{bmatrix} \mathbf{1} & \mathbf{Y} & \mathbf{Y}^{P_2} & \cdots & \mathbf{U} & \mathbf{U}^{P_2} & \cdots \end{bmatrix}.
$$

Here, higher polynomials are denoted as $\mathbf{Y}^{P_2}, \mathbf{Y}^{P_3}$, etc. For example, $\mathbf{Y}^{P_2}$ denotes the quadratic nonlinearities in the state variable Y, given by:

$$
\mathbf{Y}^{P_2} = \begin{bmatrix} \mathbf{y}_1^2(1) & \mathbf{y}_1(1)\mathbf{y}_2(1) & \cdots & \mathbf{y}_n^2(1) \\ \mathbf{y}_1^2(2) & \mathbf{y}_1(2)\mathbf{y}_2(2) & \cdots & \mathbf{y}_n^2(2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{y}_1^2(M) & \mathbf{y}_1(M)\mathbf{y}_2(M) & \cdots & \mathbf{y}_n^2(M) \end{bmatrix}.
$$

Basically, each column of $\mathbf{\Phi}(\mathbf{Y}, \mathbf{U})$ represents a candidate function for a nonlinearity in $\mathbf{f}$. The library of functions may be very large. However, since only a very small number of these nonlinearities appear in each row of $\mathbf{\Phi}(\mathbf{Y}, \mathbf{U})$, we can set up a sparse regression problem to determine the sparse matrices of coefficients $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_n \end{bmatrix}$, where $\mathbf{w}_i \in \mathbb{R}^{P \times 1}$ and P is the total number of candidate functions in the library. The nonzero elements in $\mathbf{W}$

determine which nonlinearities are active [10, 22] and the corresponding parameters. Let

$$\bar{\mathbf{Y}} \triangleq \begin{bmatrix} \mathbf{y}_1(2) & \dots & \mathbf{y}_n(2) \\ \mathbf{y}_1(3) & \dots & \mathbf{y}_n(3) \\ \vdots & \ddots & \vdots \\ \mathbf{y}_1(M+1) & \dots & \mathbf{y}_n(M+1) \end{bmatrix}.$$

This results in the overall model $\bar{\mathbf{Y}} = \mathbf{\Phi}(\mathbf{Y}, \mathbf{U})\mathbf{W} + \boldsymbol{\xi}$, where $\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\xi}_1 & \boldsymbol{\xi}_2 & \dots & \boldsymbol{\xi}_n \end{bmatrix}$ and $\boldsymbol{\xi}_i \in \mathbb{R}^{M \times 1}$ is zero-mean i.i.d. Gaussian noise with covariance matrix $\lambda\mathbf{I}$, for some $\lambda \geq 0$. The work in [10], developed methods based on Sparse Bayesian Learning for identifying each $\mathbf{w}_i$ in the above equation as the following optimization:

$$\mathbf{w}_i^* = \arg\min_{\mathbf{w}_i} \|\bar{\mathbf{y}}_i - \mathbf{\Phi}\mathbf{w}_i\|_{\ell_2}^2 + \lambda\|\mathbf{w}_i\|_{\ell_1}. \tag{2}$$

Inferring Sub-systems

When $K > 1$, we can use a similar formulation as above. However, the outstanding challenge is that there is no single $\mathbf{W}$ typically fits all the data due to the hybrid nature of the dynamical system. In addition, we have no information about which data point belongs to which subsystem. Next, we introduce a new method to tackle such a challenge.

Define $\mathbf{Z} = \bar{\mathbf{Y}} - \mathbf{\Phi}\mathbf{W} - \boldsymbol{\xi}$, where $\boldsymbol{\xi}$ is defined similarly as realizations of zero-mean i.i.d. Gaussian measurement noise with covariance matrix $\lambda\mathbf{I}$. The goal is to find a $\mathbf{Z}^* \triangleq \begin{bmatrix} \mathbf{z}_1^* & \mathbf{z}_2^* & \dots & \mathbf{z}_n^* \end{bmatrix} \triangleq \bar{\mathbf{Y}} - \mathbf{\Phi}\mathbf{W}^* - \boldsymbol{\xi}$ as sparse as possible, i.e.,

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \sum_{i=1}^{n} \|\mathbf{z}_i\|_{\ell_0} \tag{3}$$
$$\text{subject to: } \mathbf{Z} = \bar{\mathbf{Y}} - \mathbf{\Phi}\mathbf{W} - \boldsymbol{\xi}.$$

Correspondingly, we have $\mathbf{z}_i^* = \bar{\mathbf{y}}_i - \mathbf{\Phi}\mathbf{w}_i^*$. The interpretation of this optimization is to find a $\mathbf{W}$ (or equivalently a subsystem) that fits most of the input-output data. As a result, the indexes of the zero entries of $\mathbf{Z}^*$ correspond to the indexes for input-output that can be fitted by a single subsystem. This initial idea was similar to those presented in [23] for noiseless switching subsystem identification, yet we now extend this idea to a robust Bayesian algorithm that works well for noisy data (for detailed comparison, please refer to Section). To solve Eq. (3), assume, without loss of generality, that the dictionary matrix

49

$\mathbf{\Phi}$ is full rank. Define a transformation matrix $\mathbf{\Theta}$ in which each column spans the left null space of $\mathbf{\Phi}$. Then, it follows that $\mathbf{\Theta}\bar{\mathbf{Y}} = \mathbf{\Theta}Z + \mathbf{\Theta}\boldsymbol{\xi}$. Using standard maximum likelihood estimate and an appropriate Lagrange multiplier $\frac{1}{2\lambda_{\mathbf{z}}}$, we now can rewrite the above problem as an unconstrained minimization:

$$\min_{\mathbf{Z}} \frac{1}{2}\|(\tilde{\bar{\mathbf{Y}}} - \mathbf{\Theta}\mathbf{Z})^T\mathbf{\Pi}^{-1}(\tilde{\bar{\mathbf{Y}}} - \mathbf{\Theta}\mathbf{Z})\|_F^2 + \lambda_{\mathbf{z}}\sum_{i=1}^{n}\|\mathbf{z}_i\|_{\ell_0}, \tag{4}$$

where $\tilde{\bar{\mathbf{Y}}} \triangleq \mathbf{\Theta}\bar{\mathbf{Y}}$ and $\mathbf{\Pi} = \mathbf{\Theta}\mathbf{\Theta}^T$.

Remark 1 *This is the key step in the later proposed algorithm; there is no $\mathbf{W}$ in this optimization after the transformation. Instead, we are optimizing over the residual $\mathbf{Z}$.*

However, this problem is known to be computationally expensive. Instead, we use the following convex relaxation

$$\mathbf{Z}^* = \arg\min_{\mathbf{Z}} \frac{1}{2}\|(\tilde{\bar{\mathbf{Y}}} - \mathbf{\Theta}\mathbf{Z})^T\mathbf{\Pi}^{-1}(\tilde{\bar{\mathbf{Y}}} - \mathbf{\Theta}\mathbf{Z})\|_F^2 + \lambda_{\mathbf{z}}\sum_{i=1}^{n}\|\mathbf{Z}_i\|_{\ell_1}.$$

We can decompose the above optimization to a number of smaller optimizations: for $i = 1, \ldots, n$

$$\mathbf{z}_i^* = \arg\min_{\mathbf{z}_i} \frac{1}{2}(\tilde{\bar{\mathbf{y}}}_i - \mathbf{\Theta}\mathbf{z}_i)^T\mathbf{\Pi}^{-1}(\tilde{\bar{\mathbf{y}}}_i - \mathbf{\Theta}\mathbf{z}_i) + \lambda_{\mathbf{z}}\|\mathbf{z}_i\|_{\ell_1}. \tag{5}$$

Remark 2 *Specifically, we used a Bayesian formulation to replace the optimizations in Eq.* (5) *to achieve better empirical performance as detailed in the main text.*

Once this problem is solved, we consider the index set $\text{seq} = \{j \mid |\mathbf{z}_i^*[j]| \leq \epsilon_{\mathbf{z}}\}$ and further identify the sparse coefficients $\mathbf{w}_i^*$ using the following optimization

$$\mathbf{w}_i^* = \arg\min_{\mathbf{w}_i} \frac{1}{2}\|\bar{\mathbf{Y}}[\text{seq}, i] - \mathbf{\Phi}[\text{seq}, :]\mathbf{w}_i\|_{\ell_2}^2 + \lambda_{\mathbf{w}}\|\mathbf{w}_i\|_{\ell_1}.$$

The variables $\mathbf{w}_i^*$ are the coefficients of the identified subsystem.

Remark 3 *The reason to enforce $\mathbf{w}_i^*$ to be sparse is due to the constructed dictionary matrix $\mathbf{\Phi}$ usually has extra terms that are not in the true dynamics.*

We further define $\text{error} = \text{abs}(\bar{\mathbf{y}}_i - \mathbf{\Phi}\mathbf{w}_i^*)$ (here abs is an elementary-wise operator which returns the absolute value of every element of a vector) and we set the jth element of $\bar{\mathbf{y}}_i$: $\bar{\mathbf{Y}}[j, i] = 0$ and the jth row of $\mathbf{\Theta}$: $\mathbf{\Theta}[j, :] = 0$ if the jth element of error is less than $\epsilon_{\mathbf{w}}$, for some small $\epsilon_{\mathbf{w}} > 0$. This removes the data that has already been fitted by the subsystem. Once we have the new $\bar{\mathbf{Y}}$ and $\mathbf{\Theta}$, we can solve the same problem with the remaining

time points (where the corresponding elements of $\bar{\mathbf{Y}}$ and the corresponding row of $\boldsymbol{\Theta}$ are nonzero) using the exact same procedure. The number of iterations gives the minimum number of subsystems. The proposed algorithm is summarized in Supplementary Algorithm 2. The code implementation is available at https://github.com/HAIRLAB/CPSid with User's Manual in Appendix . In what follows, we shall briefly discuss extensions and variants of Supplementary Algorithm 2, which can empirically improve the performance of IHYDE.

Algorithm 2 Sub-systems Identification Algorithm

1: **Input:** Collect input-output data $\mathbf{u}(t)$ and $\mathbf{y}(t)$ for $t = 1, 2, \ldots, M + 1$. Two pre-specified thresholds $\epsilon_{\mathbf{z}}$ and $\epsilon_{\mathbf{w}}$, two tuning parameters $\lambda_{\mathbf{z}}$ and $\lambda_{\mathbf{w}}$, the upper bound of the number of subsystems $K_{\max}$

2: **Output:** Return $\{\mathbf{W}^i\}$ for $i = 1, \ldots, K$ and the number of subsystems K

3: Construct dictionary matrix $\mathbf{\Phi}(\mathbf{Y}, \mathbf{U})$ based on prior knowledge of the system

4: **for** $j = 1, \ldots, n$ **do**

5:     **for** $i = 1, \ldots, K_{\max}$ **do**

6:         Compute $\mathbf{\Theta}$ in which all column span the left null space of $\mathbf{\Theta}$: $\mathbf{\Theta\Phi} = \mathbf{0}$

7:         Solve for $\mathbf{z}_j^i$ from Algorithm 1

8:         **if** $\mathbf{z}_j^i = \mathbf{0}$ **then**

9:             $K = i$, Break

10:         **end if**

11:         $h = 1$ and seq $= [\,]$

12:         **for** $l = 1 \ldots, M$ **do**

13:             **if** the lth element of $\mathbf{z}_j^i$, i.e., $abs(\mathbf{z}_j^i[l]) \leq \epsilon_{\mathbf{z}}$ **then**

14:                 Set seq$[h] = l$ and $h \to h + 1$

15:             **end if**

16:         **end for**

17:         Solve the following convex optimization

$$\mathbf{w}_j^i = \arg\min_{\mathbf{w}_j} \frac{1}{2} \|\bar{\mathbf{Y}}[\text{seq}, j] - \mathbf{\Phi}[\text{seq}, :]\mathbf{w}_j\|_{\ell_2}^2 + \lambda_{\mathbf{w}} \|\mathbf{w}_j\|_{\ell_1} \tag{6}$$

18:         error $= \text{abs}(\bar{\mathbf{Y}}[:, j] - \mathbf{\Phi}\mathbf{w}_j^i)$

19:         **for** $l = 1 \ldots, M$ **do**

20:             **if** the lth element of error, i.e., error$[l] \leq \epsilon_{\mathbf{w}}$ **then**

21:                 Set $\bar{\mathbf{Y}}[l, j] = 0$ and $\mathbf{\Theta}[l, :] = 0$

22:             **end if**

23:         **end for**

24:     **end for**

25: **end for**

26: Return nonzero $\mathbf{W}^i \triangleq \begin{bmatrix} \mathbf{w}_1^i & \cdots & \mathbf{w}_n^i \end{bmatrix}$ for $i = 1, \ldots, K$ and the number of subsystems K

Remark 4 *When there is only one subsystem, we show that $\mathbf{Z}^i$ should be a zero matrix from the first optimization in Eq. (5). Eq. (6) should be the same as Eq. (2) since $seq = \{1, 2, \ldots, M+1\}$, which recovers the results for time-invariant nonlinear system identification in [10, 22]. As a result, IHYDE provides a unified point of view to the subsystem identification problem for any $K \in \{1, 2, \ldots\}$.*

Comparison to [23]

Bako proposed the first algorithm that novelly uses sparsity for identifying the switching systems [23]. A general framework is proposed for noiseless setting and Sec 3.4 of [23] suggests two ways to deal with the identification of subsystems with noisy data. The first method in [23] sets up an upper bound for the noise, and therefore this method is not practical for Gaussian noise as we considered in this paper. The second method in [23] formulates an optimization problem that tradeoffs the residual (mismatch between data and prediction from the model) and the energy of the noise. Next, we will show that it could be viewed as a special case of our framework. What is more, our method includes several iterations that considerably improves the results, as seen in the numerical examples below.

The identification of switching linear systems can be formulated as:

$$\bar{\mathbf{y}}_i = \boldsymbol{\Phi}\mathbf{w}_i + \mathbf{z}_i + \boldsymbol{\xi}_i, \quad i = 1, \cdots, n, \tag{7}$$

where each column of $\boldsymbol{\Phi} \in \mathbb{R}^{M \times P}$ is a candidate function. Note that residual $\mathbf{z}_i$ is sparse, and $\boldsymbol{\xi}_i$ is Gaussian noise. Reference [23] searches the subsystems as follows:

$$\min_{\mathbf{z}_i, \boldsymbol{\xi}_i} \lambda\|\mathbf{z}_i\|_{\ell_1} + \frac{1}{2}\|\boldsymbol{\xi}_i\|_{\ell_2}^2. \tag{8}$$

Next, we show that, since the objective function is convex with respect to $\mathbf{w}_i$, it yields the following form (where $\boldsymbol{\Phi}^+$ is the pseudo inverse of $\boldsymbol{\Phi}$)

$$\min_{\mathbf{z}_i, \mathbf{w}_i} \lambda\|\mathbf{z}_i\|_{\ell_1} + \frac{1}{2}\|\bar{\mathbf{y}}_i - \boldsymbol{\Phi}\mathbf{w}_i - \mathbf{z}_i\|_{\ell_2}^2 \iff \min_{\mathbf{z}_i} \lambda\|\mathbf{z}_i\|_{\ell_1} + \frac{1}{2}\|(\mathbf{I} - \boldsymbol{\Phi}\boldsymbol{\Phi}^+)(\bar{\mathbf{y}}_i - \mathbf{z}_i)\|_{\ell_2}^2. \tag{9}$$

Let $\mathbf{Q} \triangleq \mathbf{I} - \boldsymbol{\Phi}\boldsymbol{\Phi}^+$, and $rank(\boldsymbol{\Phi}) = k \, (k \leq \min(M, P))$. Using singular value decomposition, $\boldsymbol{\Phi}$ can be written as follows:

$$\boldsymbol{\Phi} = \mathbf{A}\mathbf{S}\mathbf{V}^T = \begin{bmatrix} \mathbf{A}_{1\,M \times k} & \mathbf{A}_{2\,M \times (M-k)} \end{bmatrix} \begin{bmatrix} \mathbf{S}_{1\,k \times k} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{V}_{1\,k \times P}^T \\ \mathbf{V}_{2\,(P-k) \times P}^T \end{bmatrix} = \mathbf{A}_1\mathbf{S}_1\mathbf{V}_1^T,$$

Therefore, the explicit form of $\mathbf{\Phi}^+$ is $\mathbf{\Phi}^+ = \mathbf{V}_1\mathbf{S}_1^{-1}\mathbf{A}_1^T$. Since $\mathbf{A}$ and $\mathbf{V}$ are unitary matrices, one has

$$\mathbf{Q} = \mathbf{I}_M - \mathbf{A}_1\mathbf{S}_1\mathbf{V}_1^T\mathbf{V}_1\mathbf{S}_1^{-1}\mathbf{A}_1^T = \mathbf{A}_2\mathbf{A}_2^T.$$

Remark 5 *Rather than having the derivation we had above, [23] gives the orthogonal projection matrix as,* $\mathbf{Q} = \mathbf{I}_M - \mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T$. *Note that it is only true by assuming that* $\mathbf{\Phi}$ *has full rank (which is usually not the case in our numerical examples).*

One can rewrite Eq. (9) as,

$$\min_{\mathbf{z}_i} \lambda\|\mathbf{z}_i\|_{\ell_1} + \frac{1}{2}\|\mathbf{Q}(\bar{\mathbf{y}}_i - \mathbf{z}_i)\|_{\ell_2}^2 \iff \min_{\mathbf{z}_i} \lambda\|\mathbf{z}_i\|_{\ell_1} + \frac{1}{2}(\bar{\mathbf{y}}_i - \mathbf{z}_i)^T\mathbf{A}_2\mathbf{A}_2^T(\bar{\mathbf{y}}_i - \mathbf{z}_i).$$

In contrast, IHYDE is based on Bayesian calculus. As stated before that the transformation matrix $\mathbf{\Theta}$ is the orthogonal left null space of matrix $\mathbf{\Phi}$, namely, $\mathbf{\Theta}\mathbf{\Phi} = \mathbf{0}$. Left multiply Eq. (7) by matrix $\mathbf{\Theta}$ gives

$$\mathbf{\Theta}\bar{\mathbf{y}}_i \triangleq \tilde{\bar{\mathbf{y}}}_i = \mathbf{\Theta}\mathbf{z}_i + \mathbf{\Theta}\boldsymbol{\xi}_i.$$

To get an estimate of $\mathbf{z}_i$, we use Bayesian modeling to treat all unknowns as stochastic variables with certain probability distributions. Given the characteristics of the noise $\boldsymbol{\xi}_i$, $\mathbf{\Theta}\boldsymbol{\xi}_i$ is Gaussian distributed with covariance matrix $\lambda\mathbf{\Theta}\mathbf{\Theta}^T$, i.e., $\mathbf{\Theta}\boldsymbol{\xi}_i \sim \mathcal{N}(\mathbf{0}, \lambda\mathbf{\Theta}\mathbf{\Theta}^T)$. In such a case, using the properties of Gaussian distributions, the likelihood of the output $\tilde{\bar{\mathbf{y}}}_i$ given the parameter $\mathbf{z}_i$ is

$$p(\tilde{\bar{\mathbf{y}}}_i|\mathbf{z}_i) = \mathcal{N}(\tilde{\bar{\mathbf{y}}}_i|\mathbf{\Theta}\mathbf{z}_i, \lambda\mathbf{\Theta}\mathbf{\Theta}^T) \propto \exp\left[-\frac{1}{2\lambda}(\tilde{\bar{\mathbf{y}}}_i - \mathbf{\Theta}\mathbf{z}_i)^T(\mathbf{\Theta}\mathbf{\Theta}^T)^{-1}(\tilde{\bar{\mathbf{y}}}_i - \mathbf{\Theta}\mathbf{z}_i)\right]. \tag{10}$$

Hence, using maximum likelihood estimation, we have the following optimization

$$\mathbf{z}_i = \arg\min_{\mathbf{z}_i}(\tilde{\bar{\mathbf{y}}}_i - \mathbf{z}_i)^T\mathbf{\Theta}^T(\mathbf{\Theta}\mathbf{\Theta}^T)^{-1}\mathbf{\Theta}(\tilde{\bar{\mathbf{y}}}_i - \mathbf{z}_i). \tag{11}$$

Next, we introduce sparse priors. In Bayesian models, a prior density $p(\mathbf{z}_i)$ is defined as $p(\mathbf{z}_i) = \prod_{j=1}^M p(\mathbf{Z}[j,i])$. Then, we can look at the first iteration, which yields

$$\mathbf{z}_i = \arg\min_{\mathbf{z}_i} \lambda\|\mathbf{z}_i\|_{\ell_1} + \frac{1}{2}(\tilde{\bar{\mathbf{y}}}_i - \mathbf{z}_i)^T\mathbf{\Theta}^T(\mathbf{\Theta}\mathbf{\Theta}^T)^{-1}\mathbf{\Theta}(\tilde{\bar{\mathbf{y}}}_i - \mathbf{z}_i). \tag{12}$$

One can get the orthogonal left null space matrix $\mathbf{\Theta} = \mathbf{A}_2^T$ using singular value decomposition. Therefore, Eq. (12) can be rewritten as,

$$\min_{\mathbf{z}_i} \lambda\|\mathbf{z}_i\|_{\ell_1} + \frac{1}{2}(\tilde{\bar{\mathbf{y}}}_i - \mathbf{z}_i)^T\mathbf{A}_2(\mathbf{A}_2^T\mathbf{A}_2)^{-1}\mathbf{A}_2^T(\tilde{\bar{\mathbf{y}}}_i - \mathbf{z}_i)$$

$$\iff \min_{\mathbf{z}_i} \lambda\|\mathbf{z}_i\|_{\ell_1} + \frac{1}{2}(\tilde{\bar{\mathbf{y}}}_i - \mathbf{z}_i)^T\mathbf{A}_2\mathbf{A}_2^T(\tilde{\bar{\mathbf{y}}}_i - \mathbf{z}_i). \tag{13}$$

54

Note that our method includes the methods in [23] as special cases, even with only one iteration.

Next, we illustrate the performance differences between the second method in [23] and IHYDE on the Continuous Hysteresis Loop data, with 6% noise. For both methods, 500 samples are used for training. Our method consists of an iterative algorithm; we are going to compare two scenarios. First, we set the number of iterations in our method to 1, and carefully tune the parameters for a fair comparison. Second, we set the number of iterations to 5 (which is a default setting for IHYDE). Supplementary Table 1 shows the parameters and summarizes the identification results.

It can be seen that both method with the number of Iteration =1 efficiently distinguish the number of subsystems. In theory, these two methods should have equivalent performance. However, in practice, our present method identifies the sparse dynamics describing two subsystems; while the method in [23] does not. The reason for this is due to the fact that Eq. (8) of [23] optimizes over two variables $\mathbf{z}_i, \boldsymbol{\xi}_i$. While our method in Eq. (12) optimizes over one variable $\mathbf{z}_i$. Since these optimizations are solved by first-order optimization methods, such as variants of gradient descent, such searches are prone to stuck in local minimum with larger number of variables.

More importantly, our method with 5 iterations (default setup) accurately identifies the dynamics describing two subsystems, showing an improved performance. In addition, the methods in [23] do not recover the transition rules, while IHYDE does.

Inferring Transition Logic

Once the subsystems have been identified, we can assign every input-output data point $(\mathbf{u}(t), \mathbf{y}(t))$ to a specific subsystem as shown in Supplementary Figure 1. The next step is to identify the transition logic between different subsystems. We first convert the problem of identifying the transition logic to a standard sparse logistic regression problem which can be efficiently solved by many methods in the literature. The scheme is illustrated in Supplementary Figure 2.

To proceed, we define $\eta_i(t)$ as the set membership which equals to 1 only if the subsystem i is active at discrete-time t or otherwise it equals to 0. The goal is to identify the transition rules $\mathcal{T}_{i \to j}$ between any subsystems i, j. These functions are known from

55

the information in the subsystem identification above. Define also step(x), which equals 1 if $x \geq 0$, and 0 otherwise. Mathematically, we are searching for a nonlinear function g, such that step$(g(\mathbf{y}(t), \mathbf{u}(t)))$ specifies the membership. Due to non-differentiability of step functions at 0, we alternatively relax the step function to a sigmoid function, i.e., $\eta_j(t+1) \approx \frac{1}{1+e^{-g(\mathbf{y}(t),\mathbf{u}(t))}}$, where j is a potential subsystem that we can jump to at time $t+1$. Assuming we are in subsystem i at time t, the fitness function to jump to subsystem j at time $t + 1$ is then

$$\sum_{t=1}^{M-1} \eta_i(t) \left\| \eta_j(t+1) - \frac{1}{1 + e^{-g(\mathbf{y}(t),\mathbf{u}(t))}} \right\|_{\ell_2}^2. \tag{14}$$

To solve the optimization in (14), we can parameterize $g(\mathbf{y}(t), \mathbf{u}(t))$ as a linear combination of over-determined dictionary matrix, i.e., $g(\mathbf{y}(t), \mathbf{u}(t)) \triangleq \mathbf{\Psi}(\mathbf{y}(t), \mathbf{u}(t))\mathbf{v}$, in which $\mathbf{\Psi}$ can be constructed similarly as $\mathbf{\Phi}$ in the previous section and v is a vector of to-be-discovered parameters. The cost function only takes non-zero value when $\eta_i(t) = 1$. Let $\mathcal{D} \triangleq \{t | \eta_i(t) = 1, t = 1, \cdots, M-1\}$, then

$$\sum_{t=1}^{M-1} \eta_i(t) \left\| \eta_j(t+1) - \frac{1}{1 + e^{-g(\mathbf{y}(t),\mathbf{u}(t))}} \right\|_{\ell_2}^2 = \sum_{k \in \mathcal{D}} \left\| \eta_j(k+1) - \frac{1}{1 + e^{-\mathbf{\Psi v}}} \right\|_{\ell_2}^2. \tag{15}$$

After this transformation, the minimization of Eq. (15) is known as the logistic regression. Hence, we can use standard gradient descent method to solve the logistic regression [6].

Similarly, we can also add an ℓ_1 regularizer in the optimization, i.e., we minimize the following expression

$$\sum_{t \in \mathcal{D}} \left\| \eta_j(t+1) - \frac{1}{1 + e^{-\mathbf{\Psi v}}} \right\|_{\ell_2}^2 + \beta \|\mathbf{v}\|_{\ell_1}, \tag{16}$$

where β is a predefined parameter. There are many Matlab codes for sparse linear logistic regression. Here, we adopt the implementation framework proposed in [11].

Algorithm 3 Transition Logic Identification Algorithm
_____
1: **Input:** Input-output data $\mathbf{y}(t), \mathbf{u}(t)$ and $\eta_i(t)$, $i = 1, 2 \ldots, K$ and $t = 1, 2, \ldots, M$

2: **Output:** Transition logic $\mathcal{T}_{i \to j}(\mathbf{y}(t), \mathbf{u}(t))$ for any pair i, j

3: **for** $i = 1, \ldots, K$ **do**

4:      **for** $j \neq i$ **do**

5:          Construct the dictionary matrix $\mathbf{\Psi}$ from prior knowledge as described in the main text

6:          The solution to the logistic regression in Eq. (16) gives the transition model for $\mathcal{T}_{i \to j}$

7:      **end for**

8: **end for**

9: Return all transition logic mapping $\mathcal{T}$
_____

Principles for Parameter Tuning

We tune hyperparameters based on minimum error principle described below. For a set of determined parameters $\lambda_{\mathbf{z}}, \lambda_{\mathbf{w}}, \epsilon_{\mathbf{z}}, \epsilon_{\mathbf{w}}$, we compute the fitting error for each subsystem on test data points using an *AIC*-type criteria given by

$$\text{err} = 2k + 2 \sum_{i=1}^{M} \min_{j \in \{1, \cdots, K\}} \| \bar{\mathbf{Y}}[i, :] - \mathbf{\Phi}[i, :] \mathbf{W}^j \|_2^2, \tag{17}$$

where k represents the number of non-zeros terms in all identified subsystems, and K represents the number of identified subsystems. This is a similar equation to Eq. (4).

To search for hyperparameters, we empirically set an initial grid and search the optimal hyperparameters, which is the one with the minimum AIC-type error. The initial grid is divided into 210 combinations as follows:

$$\lambda_{\mathbf{z}} \in \{10^{-7+m} | m = 1, 2, 3, 4, 5, 6, 7\},$$
$$\lambda_{\mathbf{w}} \in \{10^{-4+m} | m = 1, 2, 3\},$$
$$\epsilon_{\mathbf{z}} \in \{10^{-5+m} | m = 1, 2\},$$
$$\epsilon_{\mathbf{w}} \in \{0.001 m \| \bar{\mathbf{Y}}[\text{tseq}, i] \|_2 | m = 1, 3, 5, 7, 9\}.$$

Next, we detail the steps of this test strategy and illustrate it on four examples. The data contains 4000 samples with 6% noise. Of those, 500 samples, denoted by tseq $= \{1, 2, \ldots, 500\}$, are used for training, and all the samples, denoted by vseq $= \{1, 2, \ldots, 4000\}$, are used for

testing. Supplementary Table 2 summarizes the optimal hyperparameters and the identified subsystems obtained from the proposed minimum error principle. These examples illustrate that this initial grid and the minimum error principle can be useful for hyper-parameter tuning. Indeed, the identified parameters of all subsystem are good approximations of the true model. Note that, in general, this initial grid can be extended until the algorithm achieves good performance (low residuals).

SUPPLEMENTARY EXAMPLES

This section applies IHYDE to a number of examples ranging from power systems to robotics, showcasing the wide range of applicability of the proposed IHYDE method. We will test the IHYDE on nine different applications. The data structure of each dataset is shown in Supplementary Table 3.

Example 1: Hysteresis Relay

One of the most common Cyber Physical Systems is the Hysteresis Relay. It is found, for example, in almost all thermostats: the heater is turned on when the temperature is below a threshold, and turned off when the temperature is above another threshold. Typically, the low and high temperature switching are different to avoid frequent switching, which could damage the system. The Hysteresis Relay can be found in physical, chemical, engineering and biological applications.

The datasets for discovery are generated by Ly et. al. in [24]. The additive noise level varies from 0% to 6% in 2% increments, i.e., $N_p = \frac{\sigma_{\text{noise}}}{\sigma_y} \times 100\%$, where σ_{noise} is the noise variance and σ_y is the variance of the measurement. We apply the proposed IHYDE to data generated by an unknown Hysteresis Relay to discover its hybrid dynamical model (shown in Supplementary Figure 3a). Supplementary Table 4 shows the detailed information about this data. The discovered systems are shown in Supplementary Table 5 and Supplementary Table 6 using 2000 data-points respectively.

Using IHYDE for subsystems identification, we are able to successfully identify that there are only two subsystems that generate the datasets. In addition, the two identified subsystems are consistent with or close to the true ones from both noiseless and noisy data.

Specifically, with or without redundant dictionary functions, we are able to identify the true systems, achieving very similar discovery results. This, in other words, demonstrates that the IHYDE is able to discover the true subsystems, the number of subsystems together with parameterizations of every subsystem.

Once all subsystems have been identified and all data points have been classified, IHYDE identifies the transition logic between subsystems. When there is redundant dictionary function (i.e., when prior knowledge is available about the structure of transition logic), IHYDE is able to precisely identify the correct transition logic. The identified results are shown in Supplementary Table 7. When there exists redundant dictionary functions, IHYDE still successfully identifies the transition logic. The identified results are shown in Supplementary Table 8.

Example 2: Continuous Hysteresis Loop

A Continuous Hysteresis Loop is yet another classical hybrid system– here we use the Preisach model [24] for data simulation. In our setup, each subsystem has its own input-output behavior while the transitions occur when the input hits certain thresholds as shown in Supplementary Figure 3b. The detailed information is summarized in Supplementary Table 9. We apply the IHYDE to reverse engineering the Continuous Hysteresis Loop using 2000 data points generated by [24].

The identified systems are shown in Supplementary Table 10 and Supplementary Table 11. In contrast with the previous Hysteresis Relay example, the IHYDE will obtain false classification results as the noise level increases. Yet, IHYDE is still able to identify the actual subsystem dynamics up to some precision.

Once all subsystems have been identified and all data points have been classified, IHYDE identifies the transition logic between subsystems. Even when there exists redundant dictionary function, IHYDE is able to precisely identify the correct transition logic. The identified results are shown Supplementary Table 12.

Example 3: Phototaxis Robot

Consider a Phototaxis Robot with a hybrid dynamical system model shown in Supplementary Figure 3c [13], the robot has phototaxis movement: it approaches, avoids, or remains stationary depending on the color of light. As described in [24], the output y is velocity of the robot. There are five inputs: u_1 and u_2 are the absolute positions of the robot and the light, respectively, while $\{u_3, u_4, u_5\}$ is a binary, one-hot encoding of the light color, where 0 indicates the light is off and 1 indicates the light is on.

Similar to previous examples, 2000 data points are used. The detailed information is shown in Supplementary Table 13 and Supplementary Table 14. IHYDE will obtain false classification results as the noise level increases (shown in Supplementary Table 15 and in Supplementary Table 16). Yet, IHYDE is still able to identify the actual subsystem dynamics without redundant dictionary functions when noise intensity is low. When there exists redundant dictionary functions, IHYDE can identify all the subsystems when there is no noise. When noise level increases, IHYDE still identifies the right number of subsystems, yet the third identified subsystem is different from the true one, i.e., $y = 0$.

Again, once all subsystems have been identified and all data points have been classified, IHYDE identifies the transition logic between subsystems. IHYDE is able to precisely identify the correct transition logic both when there is no redundant dictionary function (Supplementary Table 17) and when there is (Supplementary Table 18). At a first glance, the inferred transition logic is different from the actual ones. Given u_3, u_4, u_5 are binary values, still, the inferred transition logic are equivalent to the actual ones.

Example 4: Non-linear Hybrid System

Consider the Nonlinear Hybrid System shown in Supplementary Figure 3d. This example is a system without any physical counterpart, yet it is useful to evaluate the capabilities of IHYDE for finding non-linear expressions. The system consists of three subsystems, where all of the behaviors and transition logic consist of non-linear equations which cannot be modeled via parametric regression. All the expressions are a function of the variables u_1 and u_2, the discriminant functions are not linearly separable and the transitions are modally dependent.

The detailed information for this system is summarized in Supplementary Table 19 and Supplementary Table 20. Using 2000 data points in dataset generated by [24], the identified results are shown in Supplementary Table 21 and Supplementary Table 22. Using IHYDE for subsystems identification, we are able to successfully identify that there are three subsystems that generate the datasets. In addition, the three identified subsystems are consistent with or close to the true ones from both noiseless and noisy data. IHYDE is also able to precisely identify the correct transition logic both when there is no redundant dictionary function (Supplementary Table 23) and when there is (Table 24).

Example 5: Autonomous Car

This example presents the results of IHYDE applying to an autonomous car built in our lab. The autonomous car consists of a body, a MK60t board, a servo motor, tow driving motors and a camera. During execution, the embedded camera captures the upcoming road layouts to check whether there is an upcoming straightaway or curve. Naturally, the car will drive faster on straightaways and slower on the curves.

Based on this design principle, we would like to design a hybrid dynamical system with two subsystems and simple transition logic to realize this goal as shown in the right panel of Supplementary Figure 5. The car measures current speed by encoder and calculates the Δu, a control input to the motor. The speed control strategy is based on an incremental PI control algorithm, which is widely used in control systems. The incremental PI algorithm is developed from position PI algorithm. The position PI model is described as below and can be seen in Fig. Supplementary Figure 4. $r(t)$ represents the input of the whole system (the expected speed $v_{\mathrm{expect}}(t)$) and $c(t)$ represents the output of the whole system (the real speed observed $v(t)$).

In the figure, $u(t)$ is the output of the controller and it can be calculated from $e(t)$:

$$u(t) = P\left[e(t) + \frac{1}{T_{\mathrm{I}}}\int_0^t e(t)dt\right],\tag{18}$$

where P is the constant for the proportional control, T_{I} is the time constant for the integral control. In the Laplace domain, Eq. (18) is equivalent to $U(s) = D(s)E(s)$, where $U(s)$ and $E(s)$ are the Laplace transform of $u(t)$ and $e(t)$ respectively, $D(s)$ represents the transfer

61

function of the controller:

$$D(s) = \frac{U(s)}{E(s)} = P\left(1 + \frac{1}{T_{\mathrm{I}}s}\right). \tag{19}$$

Since the controller is implemented by a computer, it must be first converted to discrete time. The integral can be approximated by

$$\int_0^t e(t)dt \approx \sum_{i=0}^k Te(i) \Rightarrow \frac{de(t)}{dt} \approx \frac{e(k) - e(k-1)}{T}. \tag{20}$$

So we obtain the following control law

$$u(k) = P\left[e(k) + \frac{T}{T_{\mathrm{I}}}\sum_{i=0}^k e(i)\right]. \tag{21}$$

The position PI algorithm is usually approximated by an incremental PI algorithm:

$$\Delta u(k) \triangleq u(k) - u(k-1) = P[e(k) - e(k-1)] + Ie(k), \tag{22}$$

where $I \triangleq \frac{PT}{T_I}$. In the autonomous car example, we have

$$r(k) = v_{\mathrm{expect}}(k), \; c(k) = v(k), \; e(k) = v_{\mathrm{expect}}(k) - v(k). \tag{23}$$

Here $v_{\mathrm{expect}}(k)$ is the expected velocity depending on whether there is an upcoming straight-away or curve from the camera. We set up a faster velocity on straightaways and slower one on the curves. Substituting $e(k)$ into the Eq. 22, we obtain

$$\Delta u(k) = P[v(k-1) - v(k)] + P[v_{\mathrm{expect}}(k) - v_{\mathrm{expect}}(k-1)] + I[v_{\mathrm{expect}}(k) - v(k)]. \tag{24}$$

When the car changes its expected velocity, this could lead to a more complicated hybrid dynamical system than we would design as shown in Supplementary Figure 5. This side effect is due to the abrupt switching and discretization. In practice, we normally neglect these subsystems in the modeling, analysis and design. The flow chart of the PI control algorithm is shown in Supplementary Figure 6.

Next, we demonstrate how IHYDE can help in the design process. In the first experiment, the autonomous car failed to drive through the track. We collected the experimental data and used IHYDE to discover the failed system. We compared the discovered system model with the to-be designed one and found an implementation error that led the system to failure. We expected a higher speed when the car is running in a straight line and a lower speed while

it is running on a curve. The model from the failed experiments showed that the transition logistics should be reversed as shown in Supplementary Figure 7 and Supplementary Table 26. We fixed the bug and as a result the autonomous car was able to run through the track. Finally, as a validation, we collected the data shown in Supplementary Table 25 and repeated the modeling process in Supplementary Table 27 and Supplementary Table 28.

In summary, IHYDE successfully reverse engineered the control strategy of the CPS. Additionally, we deliberately swapped the straightway and curve speeds to mimic a software bug. The modeled system immediately pinpointed the location of the faulty software and yielded important information for debugging the system.

Example 6: Chua's Circuit

In this subsection and the next one, we shall apply IHYDE to data that is obtained from experiments shown in Supplementary Table 29 and Supplementary Table 30. Chua's circuit shown in Supplementary Figure 8 is the simplest electronic circuit that exhibits classic chaotic behavior. It consists of an inductor, two capacitors, a passive resistor and an active nonlinear resistor which fits the condition for chaos with the least components. The most important active nonlinear resistor is a conceptual component and the resistor can be built with operational amplifiers and linear resistors. The current-voltage characteristics of the nonlinear resistor are plotted in Supplementary Figure 8b.

By design, the current-voltage relationship can be described as follows:

$$i(U) = \begin{cases} aU + (b-a)(U-E), \ U > E \\ aU, \qquad\qquad\qquad -E < U < E \\ aU + (b-a)(U+E), \ U < -E, \end{cases} \tag{25}$$

or equivalently

$$i(U) = bU + \frac{1}{2}(a-b)(|U+E| - |U-E|). \tag{26}$$

In both equations, a, b, E are parameters defined in Supplementary Figure 8b.

The nonlinear resistor can be built using the circuit realization as shown in Supplementary

Figure 8c. From KCL and KVL, we obtain

$$\frac{C_1 dU_1}{dt} = \frac{U_2 - U_1}{R} - i(U_1)$$
$$\frac{C_2 dU_2}{dt} = \frac{U_1 - U_2}{R} + I_L \tag{27}$$
$$-L\frac{dI_L}{dt} = U_2.$$

Then we introduce a number of variables to simplify the above equations:

$$y_1 = \frac{U_1 * \text{threshold}}{E}$$
$$y_2 = \frac{U_2 * \text{threshold}}{E} \tag{28}$$
$$y_3 = \frac{I_L * \text{threshold}}{E},$$

where the variables are defined as below:

- C_1: Capacity of Capacitor1. C_2: Capacity of Capacitor2. L: Inductance of the Inductor.

- U_1 : Voltage through Capacitor1. U_2: Voltage through Capacitor2.

- I_L: Current through Inductor. i: Current through the nonlinear resistor.

- E: the threshold voltage for the nonlinear resistor. a: the slope of low voltage for the nonlinear resistor. b: the slope of high voltage for the nonlinear resistor

- threshold: a parameter of the Chua's circuit, which equals to E in this experiment.

Let

$$\alpha = \frac{1}{RC_1}$$
$$\beta = \frac{1}{L} \tag{29}$$
$$f(y)|_{y=x\frac{threshold}{E}} = R\frac{\text{threshold}}{E}i(x),$$

we have the equation in the following form:

$$\frac{dy_1}{dt} = \alpha[y_2 - y_1 - f(y_1)]$$
$$RC_2\frac{dy_2}{dt} = y_1 - y_2 + Ry_3 \tag{30}$$
$$\frac{dy_3}{dt} = -\beta y_1,$$

64

where $f(x)$ is given by

$$f(x) = \begin{cases} k_1 x + b_1, & x < -\text{threshold} & \text{(31a)} \\ k_0 x, & -\text{threshold} < x < \text{threshold} & \text{(31b)} \\ k_1 x + b_2. & x > \text{threshold} & \text{(31c)} \end{cases}$$

and

$$\begin{aligned} k_0 &= Ra \\ k_1 &= Rb \\ b_1 &= R(a - b)\text{threshold} \\ b_2 &= R(a - b)\text{threshold}. \end{aligned} \tag{32}$$

The behavior of the system will be changed between chaos and non-chaos depending on the value of R. Each mode in Eq. (31a), Eq. (31b) and Eq. (31c), corresponds to subsystem 1, subsystem 2 and subsystem 3 respectively. We focus on the discovery of the first equation in Eq. (30) and only collect the value of y_1 and y_2. The output data from the Chua's circuit can be seen in Supplementary Figure 8d.

From the true parameters in Supplementary Table 31, we can compute the true coefficients of $f(x)$ to determine $\frac{dy_1}{dt}$:

$$10^{-5}\frac{dy_1}{dt} = \begin{cases} 1.0858y_2 - 0.2115y_1 - 0.5349, & y_1 < -1.5 & \text{(33a)} \\ 1.0858y_2 + 0.1451y_1, & -1.5 < y_1 < 1.5 & \text{(33b)} \\ 1.0858y_2 - 0.1994y_1 + 0.5168. & y_1 > 1.5. & \text{(33c)} \end{cases}$$

The algorithm accurately infers the form of Eq. (33c) from the data as shown in Supplementary Table 32.

Example 7: Monitoring of Industrial Processes

The next example illustrates how IHYDE can be used for fault detection in mechanical engineering. Experiments conducted on a wind turbine system experimental platform [19] are used to verify its effectiveness.

This system contains a power supply of 380V, an inverter, a motor, a gearbox, a power generator, and a load. The platform is used to simulate the process of air flow through wind turbines to generate electricity. Specifically, the motor with a gear reducer of $20 : 1$ ratio supplies the generator with mechanical power through the gearbox. In this experiment, we

adopt the mode of gearbox inversion to simulate the operation of wind turbine system. The gearbox has been widely used to provide speed and torque conversions from a motor to generator in wind turbines [17]. This system has a gearbox with three shafts, i.e., shaft with low speed, shaft with intermediate speed and shaft with high speed. The load consumes the power generated by the generator. We can measure the root-mean-square current and voltage of the motor from the inverter. The current of generator can be captured by oscilloscope and its voltage is measured through multimeter. We measure the voltage of the load in the same way.

We perform experiments under normal and faulty conditions. Both experiments are performed in the situation where the generator speed is 200 revolutions per minute and the load is 1.5 KNm. One-third of the tooth width cut off from the gear tooth on the high-speed shaft is considered as the faulty condition. In the normal operation, the motor power is 383.01W and the generator power is 53.28W; the load voltage is 75V in the faulty condition.

Two sets of current data are measured at the frequency of 1000Hz connected in series for identification. The first dataset contains 19,995 data points sampled under normal operating condition, the other has 20,000 data points obtained from the faulty condition. Then, we down-sample at the period of 0.3s and denote as $i(k)$ in which $k = 1, \ldots, 133$. Supplementary Table 33 shows the detailed information for this data.

As described in the main text, here we used an online monitoring scheme. We construct the output $\mathbf{y} \in R^{64 \times 1}$, including 61 current measurements from 1.8s to 19.8s in the normal condition and 3 data points from 20.1s to 20.7s in the faulty condition when the mismatch is large. Specifically

$$\mathbf{y} = \left[i(7)\ \ i(8)\ \ \ldots\ \ i(70) \right]^T .$$

With the candidate terms of the polynomial combinations of $i(k), \ldots, i(k+5)$ up to second order, we construct a dictionary matrix $\mathbf{\Phi} \in R^{64 \times 28}$ as follows:

$$\mathbf{\Phi} = \begin{bmatrix} 1 & i(6) & \cdots & i(1) & i^2(6) & \ldots & i^2(1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & i(69) & \ldots & i(64) & i^2(69) & \ldots & i^2(64) \end{bmatrix},$$

It is worth mentioning that this experiment is a one-shot experiment. However, many state of the art machine learning methods [18] need (a large number of) historical data

including its labels (healthy or faulty), while industrial data is often unlabeled and scarce. Therefore, these algorithms are not a good solution to this type of one-shot industrial problem. And, this example demonstrates the capability of IHYDE to the identification of the fault in industrial processes. Supplementary Figure 9 shows that the relative fitting error ratio is small. The identified results and details of the IHYDE are presented in Supplementary Table 34, which shows that the identified time for the fault occurrence is the same as the real fault time 68. We only use three fault points to realize the fault detection.

Example 8: Power Grid Fault Detection

The next example illustrates how IHYDE can be used in real-time monitoring applications. Consider the fault detection problem in a smart grid. The design of monitoring schemes to diagnose anomalies caused by unpredicted or sudden faults on power networks is of great importance.

Here we consider a benchmark power network, IEEE 14 bus test system. Suppose the line connecting buses 6 and 12 disconnects at time 31, changing the admittance between these two buses to zero. We simulate the data summarized in Supplementary Table 35 and only pass the data to IHYDE without other information. IHYDE can immediately detect the occurrence of this event and estimate the new admittance matrix using the next 10 measurements. It successfully discovers two different subsystems from data and pinpoints the difference in the discovered subsystems which corresponds to the fault. Given the frequency at which PMUs sample voltage and current, IHYDE is able to locate the fault in a few hundred milliseconds after the event occurs, enabling the operators to detect the event, identify its location, and take remedial actions in near real-time.

Example 9: Identification of Real-time Models for Smart Grid

This example illustrates how the proposed IHYDE method can be used to solve the identification problem in smart grid, which contains two major parts, that is, smart infrastructure system and smart management system [14]. It is crucial to obtain real-time models for smart management system to achieve resilient and efficient operations. Accurate model information is not only necessary for daily operation and scheduling, but also critical for

other advanced techniques such as state estimation and optimal power flow computation. However, such information is not always available in distribution systems due to frequent model changes. For example, the model of a distribution system connected with photovoltaic panels maybe change once every eight hours [15]. Furthermore, some unexpected events, such as line faults and unreported line maintenance, can lead to model changes. Moreover, network reconfiguration (such as switch action for balancing loads and avoiding voltage sag) happens frequently in distribution systems. Therefore, model identification in real-time is meaningful.

We apply IHYDE to identify network models in real-time and to infer transition logic for model changes using data from advanced metering infrastructure. The 33-bus benchmark distribution system [25] is used to generate data. Consider the situation where the increase of loads at some remote nodes of a feeder causes the voltage sag, an operator then takes switch action for load balancing and voltage regulation. Supplementary Figure 10 depicts the switching topologies and the real transition logic. The detailed actions and switching time are shown in Supplementary Table 38. Measurements are generated via solving nonlinear power flow equations using MATPOWER toolbox [16] in MATLAB.

Suppose that we can measure all the active and reactive power consumption, voltage magnitudes and phases of the nodes, denoted by $\mathbf{Y}$ as follows.

$$\mathbf{Y} = \begin{bmatrix} P_1(1) & Q_1(1) & U_1(1) & \theta_1(1) & \cdots & P_{33}(1) & Q_{33}(1) & U_{33}(1) & \theta_{33}(1) \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ P_1(M) & Q_1(M) & U_1(M) & \theta_1(M) & \cdots & P_{33}(M) & Q_{33}(M) & U_{33}(M) & \theta_{33}(M) \end{bmatrix}$$

where $U_i(t)$, $\theta_i(t)$, $P_i(t)$ and $Q_i(t)$ are the voltage magnitude, voltage phase, active and reactive power of Bus i at time instant t, respectively. The total sampling time M is set to 180 in the following simulation. Supplementary Table 37 shows the detailed information of this data.

For each node, we apply IHYDE to identify the responding column of the admittance matrix. The output $\mathbf{y}_i \in R^{2M \times 1}$ of Bus i is $\mathbf{y}_i = [P_i(1), Q_i(1), \cdots, P_i(M), Q_i(M)]^T$. The quadratic terms for the voltages are chosen as the dictionary function based on Ohm's law and power factor; sine and cosine terms are also considered, since there are voltage angle differences for delivering power from one bus to another bus. The jth column of dictionary

matrix $\boldsymbol{\Phi}^i \in R^{2M \times 66}$ is as follows:

$$\boldsymbol{\phi}_j = [U_i(1)U_j(1)\cos\theta_{ij}(1), U_i(1)U_j(1)\sin\theta_{ij}(1), \cdots, U_i(M)U_j(M)\cos\theta_{ij}(M), U_i(M)U_j(M)\sin\theta_{ij}(M)]^T,$$

where $\theta_{ij}(t)$ denotes the phase difference between nodal voltages of Bus i and j at time instant t.

Supplementary Table 39 shows the identified results and the detailed tuning parameters of the proposed algorithm. For example, at Bus 12, the maximum relative identification ratio of Base configuration and Changed configuration are 0.00057% and 0.00182%, respectively. The identified admittance matrices at time instants $31, 61, 91, 121, 151$ are very different from that of the previous moments, which indicates the model switching. The results demonstrate that IHYDE can identify the models accurately and pinpoint model switching time correctly. We add the difference of voltage magnitude between different times, denoted by ΔU_i, into dictionary matrix for logic identification. Supplementary Table 40 indicates that the identified logic is consistent with the real logic with small error. Specifically, the result of $\mathcal{T}_{1\to 2}$ (switching from subsystem 1 to 2) reveals that the voltage drop of node 10 at feeder 3 are more than 0.0500 at time 30, subsequently, switch action is taken to avoid sharp voltage drop. The tie switch between Bus 12 and 22 is closed, while the sectionalizing switch between Bus 11 and 12 opens. This is consistent with our preset reason that loads at Bus $9, 10, 11$ increase rapidly at time 30. There are many indistinct physical phenomenons in actual power system and IHYDE can be utilized to help engineers understand the hidden mechanism behind it.

Example 10: Discovery of Human Atrial Action Potential Models

In this section, we apply IHYDE to a human atrial action potential (AP) model proposed in [26] to show the applicability of IHYDE to the discovery in biology. The parameters of the human atrial AP model are determined based on the data that is directly measured on human atrial cells and that is from AP model of guinea pig ventricular and rabbit atrial. The AP model can reproduce a variety of observed AP behaviors and provide potential insights into its underlying ionic mechanisms. The human atrial AP and ionic currents that underlie its morphology are of great importance to our understanding and prediction of the electrical properties of atrial tissues under normal and pathological conditions.

Specifically, the cell membrane is modeled as a capacitor connected in parallel with variable resistances and batteries representing the ionic channels and driving forces. The AP model includes 21 differential equations and 163 parameters in total (see the code for detailed information). The membrane potential formulation is as follows

$$\frac{dV}{dt} = \frac{-(I_{\text{ion}} + I_{\text{st}})}{C}, \tag{34}$$

where V is membrane potential, and C is the constant total membrane capacitance. I_{ion} and I_{st} are the total ionic current and stimulus current flowing across the membrane, respectively.

Supplementary Figure 11 shows that the action potential generated by the AP model through voltage clamp method is a spike-and-dome morphology commonly observed in human atrial AP recordings. We apply the stimulation current with 2 ms pulses of 2 nA amplitude across the cell membrane every 1000 ms. To check the performance of the IHYDE method, we focus on two representative equations about gating variables h and j with time-varying parameters as follows:

$$\frac{dh}{dt} = \alpha_h - h(\alpha_h + \beta_h) \tag{35}$$

$$\frac{dj}{dt} = \alpha_j - j(\alpha_j + \beta_j), \tag{36}$$

where h and j are fast and slow inactivation gating variables for fast inward Na^+ current, respectively. For convenience, we present the time-varying parameters $\alpha_h, \alpha_j, \beta_h, \beta_j$:

$$\alpha_h = \begin{cases} \alpha_{h1} \triangleq 0.135 \exp(-\frac{V+80}{6.8}) & V < -40 \\ \alpha_{h2} \triangleq 0 & V \geq -40 \end{cases}$$

$$\alpha_j = \begin{cases} \alpha_{j1} \triangleq [-1.2714 \times 10^5 \exp(0.2444V) - \\ \qquad 3.474 \times 10^{-5} \exp(-0.04391)]\frac{V+37.78}{1+\exp[0.311(V+79.23)]} & V < -40 \\ \alpha_{j2} \triangleq 0 & V \geq -40 \end{cases}$$

$$\beta_h = \begin{cases} \beta_{h1} \triangleq 3.56 \exp(0.079V) + 3.1 \times 10^5 \exp(0.35V) & V < -40 \\ \beta_{h2} \triangleq \{0.13[1 + \exp(-\frac{V+10.66}{11.1})]\}^{-1} & V \geq -40 \end{cases}$$

$$\beta_j = \begin{cases} \beta_{j1} = 0.1212\frac{\exp(-0.01052V)}{1+\exp[-0.1378(V+40.14)]} & V < -40 \\ \beta_{j2} = 0.3\frac{\exp(-2.535 \times 10^{-7}V)}{1+\exp[-0.1(V+32)]} & V \geq -40. \end{cases}$$

When the gating variables h and j are equal to 1, the fast inward Na^{2+} current is inactive completely. Supplementary Figure 12 depicts that they gradually rise to their resting values

70

0.9775 and 0.9649 after stimulus.

It is clearly observed that membrane voltage gradually returns to its stable resting potential -81mV after the stimulation from Supplementary Figure 11. During the process, the dynamics for gating variables h and j has been switched as shown in Supplementary Figure 12 when the membrane voltage V goes through -40 mV. Supplementary Table 41 summarizes the data structure that is used for identification. We apply IHYDE to discover the different models and the transition logic only using measurements. The first-order differential values of h and j are considered as their output, respectively. For instance, we down-sample the differential value of h during $120 - 500$ ms as its output

$$\mathbf{y}_h = \left[\frac{dh(120)}{dt}, \frac{dh(120 + \delta_t)}{dt}, \cdots, \frac{dh(499.8)}{dt}\right]^T \in R^{1267 \times 1}.$$

The sampling period δ_t is set to 0.3 ms, and there are 1267 data points for each variable. The dictionary matrix of gating variables h and j, denoted by $\mathbf{\Phi}^h$ and $\mathbf{\Phi}^j$, respectively, are established based on the terms of the above equations

$$\mathbf{\Phi}^h = \begin{bmatrix} \exp(-\frac{V(t_1)+80}{6.8}) & h(t_1)\exp(-\frac{V(t_1)+80}{6.8}) & h(t_1)\beta_{h1}(t_1) & h(t_1)\beta_{h2}(t_1) \\ \vdots & \vdots & \vdots & \vdots \\ \exp(-\frac{V(t_n)+80}{6.8}) & h(t_n)\exp(-\frac{V(t_n)+80}{6.8}) & h(t_n)\beta_{h1}(t_n) & h(t_n)\beta_{h2}(t_n) \end{bmatrix},$$

$$\mathbf{\Phi}^j = \begin{bmatrix} \alpha_{j1}(t_1) & j(t_1)\alpha_{j1}(t_1) & j(t_1)\frac{\exp(-0.01052V(t_1))}{1+\exp[-0.1378(V(t_1)+40.14)]} & j(t_1)\frac{\exp(-2.535\times 10^{-7}V(t_1))}{1+\exp[-0.1(V(t_1)+32)]} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_{j1}(t_n) & j(t_n)\alpha_{j1}(t_n) & j(t_n)\frac{\exp(-0.01052V(t_n))}{1+\exp[-0.1378(V(t_n)+40.14)]} & j(t_n)\frac{\exp(-2.535\times 10^{-7}V(t_n))}{1+\exp[-0.1(V(t_n)+32)]} \end{bmatrix},$$

where t_1 and t_n are 120 and 499.8 ms, respectively.

The identified results and the detailed parameters are summarized in Supplementary Table 43. We can see that IHYDE identifies the subsystem and pinpoints the changing time correctly. The identified logic (see Supplementary Table 42 and Supplementary Table 45) for both gating variables are $V \leq -40.0093$, which is very close to the real logic $V \leq -40$.

Next, we repeat the modeling of this system with the assumption that the choice of dictionary functions is unclear and/or the domain knowledge is lacking. In such cases, we consider a canonical dictionary function, such as polynomials approximations. The results are summarized in Supplementary Table 44. IHYDE can still detect the transition points. However, the nonlinear dynamics are different than the true ones: as expected, it identifies

instead a polynomial approximation of the original nonlinear dynamics. While these dynamics can still be used for simulation and trajectory prediction, they are not in a form that reveals physical meaning. For an interpretable model, we require domain knowledge. Please see Supplementary Discussion 3 for another example on canonical dictionary functions.

Example 11: Non-hybrid Dynamical Systems

We also tested IHYDE on non-hybrid dynamical systems using datasets in [22] to illustrate the applicability of IHYDE. The details of simulation datasets in [22] are presented in Supplementary Table 46. Supplementary Table 48 shows the hyperparameters tuned for IHYDE. The results are summarized in Supplementary Table 47. Overall, IHYDE unifies previous results for the discovery of non-hybrid dynamical systems, such as examples in [22].

SUPPLEMENTARY DISCUSSIONS

The IHYDE algorithm has been tested in a number of examples. As the number of dictionary functions and the amount of noise increase, the algorithm is eventually unable to identify the actual model. Although it can fit data very well, it usually obtains more complex models than the true ones. This is actually a typical problem in system identification [21]. When the data is not informative, it leads to non-identifiability issues, i.e., there will exist multiple hybrid dynamical systems that can produce the same data, which prevents the proposed IHYDE algorithm from finding the true system.

Supplementary Discussion 1: Identifiability

Consider the following linear system with unknown parameters k_1 and k_2

$$
\frac{d}{dt} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} k_1 & 1 + k_2 \\ 0 & k_1 + k_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u
$$
$$
y = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.
$$

(37)

The observed output is plotted as follows in Supplementary Figure 13 (the system is stimulated by an impulse input, i.e., $u(t) = \delta(t)$ where $\delta(\cdot)$ is the Dirac delta function):

However, any k_1, k_2 with $k_1 + k_2 = 0.8$ produces the same input-output data. For example, the actual system

$$\frac{d}{dt} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.4 & 1.4 \\ 0 & 0.8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

$$(38)$$

and

$$\frac{d}{dt} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.3 & 1.5 \\ 0 & 0.8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$(39)$$

are indistinguishable from the input-output data alone. Hence, without more information, the true parameters cannot be identified using any methods.

Supplementary Discussion 2: Data Informativity

The previous example demonstrates that, when the parameterization is not identifiable, no algorithm is able to identify the correct parameters. Next, we shall demonstrate another example where, even though the subsystem is identifiable, the data is not informative enough. For example, some of the logic transitions never occur. Consider the following hybrid dynamical system in Supplementary Figure 14. If the system starts at initial condition $y(0) = 18$, then it always stays in subsystem 1. Hence, with the data generated, no algorithm is able to identify the complete hybrid dynamical system.

Supplementary Discussion 3: Canonical dictionary functions

With an example, this section explores the effect of dictionary functions when the right choice of dictionary functions is unclear and/or domain knowledge is lacking. Consider a

hybrid dynamical system with two subsystems: subsystem 1 follows $\dot{x} = -x^3$, and subsystem 2 with $\dot{x} = -\cos(x)$. This hybrid system switches every $0.5s$ during $t \in [0, 10]$. We set the initial condition to $x_0 = 0.99$ and the sampling period to $0.005s$. Then, 2000 simulated data points are obtained. We choose the first 1000 points as training data set, denoted by tseq $= \{1, \cdots, 1000\}$, and the whole data as testing data set. Assume there is no prior knowledge about the function forms of the subsystems. Then, pick a canonical dictionary function consisting of polynomials up to fifth order, grid the hyperparameters using the initial grid set in Supplementary Method 1, and use the minimum error principle to search a best set of hyperparameters.

Supplementary Table 49 summarizes the identified results. IHYDE first correctly discovers one of the subsystems $\dot{x} = -x^3$ and then discovers a second subsystem with the form $\dot{x} = -1 + \frac{1}{2}x^2$, which is different from the true subsystem. On the other hand, this is consistent with the Taylor series expansion of $\cos(x) = 1 - \frac{1}{2}x^2 + O(x^4)$.

SUPPLEMENTARY: USER'S MANUAL OF THE CODE

Identification of hybrid dynamical systems (IHYDE) is a open-source Matlab toolbox for automating the mechanistic modeling of hybrid dynamical systems from observed data. IHYDE has low computational complexity, enabling its application to real-world CPS problems. IHYDE implements the clustering-based algorithms described in the Data-driven Discovery of Cyber Physical Systems. It can also be used, positively, for the creation of guidelines for designing new CPSs. IHYDE uses routines of the CVX and SLR toolboxes for constructing and solving disciplined convex programs (DCPs).

Download the latest version of IHYDE toolbox in a directory and add its path (and the path of the subdirectories) to the Matlab path. The IHYDE toolbox consists of directories listed in Table 50. Table 51, Table 52, Table 53 and Table 54 contain the introduction of IHYDE's API.

To quickly familiarize with IHYDE, examples are in the directory /IHYDE/examples. These .m files can also be used as templates for other experiments. We shall use the autonomous car example to explain the code. First, we load the data.

```
addpath('./tools');
addpath('./data');
```

```matlab
basis_function.work='off';
data=load('normal_car.mat');    %% load Data
index = 1000:1400;
flag = data.flag(index);    % 1:straghtway    0:curve
dy = data.dy(index);
v = data.v(index)/10;

polyorder = 4; % The highest order of the polynomial is 4
    order
A= library(v,polyorder,memory,basis_function);%make a
    library
A = A(memory+2:end,:);
dy = dy(memory+2:end);  %dpwm_{k}
flag = flag(memory+2:end);  %flag_{k}
v_k1 = v(memory+1:end-1,:);    % v_{k-1}
v_k2 = v(memory:end-2,:);% v_{k-2}
v_k3 = v(memory-1:end-3,:);% v_{k-3}
v = v(memory+2:end,:);    % v_{k}
```

Then, we initialize the parameters and identify the systems by function `ihyde`.

```matlab
parameter.MAXITER = 5;    %the iter for  the sparsesolver
    algorithm
parameter.max_s = 20;             % the max number of
    subsystems
parameter.epsilon = [100   8];   %  the fist element in
    lambda is   epsilon_z and the second is epsilon_w
parameter.Phi = A;    % the library
parameter.y = dy;   %dpwm
parameter.normalize_y = 1;  % normalize:1      unnormalize
    :0
[result]=ihyde(parameter);       % inferring   subsystems
```

Function `ihyde` will return a preliminary identified result which contains the details of subsystems. Since we want to get a better result based on the minimum error principle, we

75

use function `finetuning` to fine-tune the results.

```
result.epsilon = parameter.epsilon(2);% use epsilon_w as
    the epsilon in finetuning
result.lambda = parameter.lambda(2);% use lambda_w as the
    epsilon in finetuning
result.threshold = [0.05];     %set a threshold for
    clustering
final_result  = finetuning(result);    % finetuning each
    subsystems
sys = final_result.sys;  % get the identified subsystems
idx_sys = final_result.idx;% get the index of each
    subsystems
```

The code for inferring transition logic between subsystems is shown below.

```
Phi2 = [ones(size(flag)) flag 1./v sin(v) cos(v) v.^2
    v_k1./v_k2 v_k3.^2 ];% library for inferring transition
    logic between subsystems.
para_log.idx_sys = idx_sys;
para_log.beta= 0.5;    % the tradeoff of l1-sparse logistic
    regression
para_log.y = dy;
para_log.Phi2 = Phi2;
[syslogic,labelMat,data] = ihydelogic(para_log);
```

The identified results are saved in `sys`, `idx_sys` and `syslogic`.

SUPPLEMENTARY REFERENCES

[1] Visintin A. *Differential Models of Hysteresis*. (Springer, 1994).

[2] Grant M. & Boyd S. CVX: Matlab software for disciplined convex programming, version 2.0 beta. `http://cvxr.com/cvx`, (2013).

[3] Candes E., Romberg J. & Tao T. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inf. Theory* **52**, 489-509 (2006).

[4] Donoho D. L. & Elad M. Optimally sparse representation in general (nonorthogonal) dictionaries via l1minimization. *Natl. Acad. Sci. USA* **100**, 2197 (2003).

[5] Hastie T. et al. *The elements of statistical learning*, (Springer, 2009).

[6] Murphy K.P. *Machine learning: A probabilistic perspective* (MIT Press, 2012).

[7] Palmer J., Wipf D., Kreutz-Delgado K. & Rao B. Variational EM algorithms for non-Gaussian latent variable models, *Advances in neural information processing systems* **18**, 1059 (2006).

[8] Bishop C. *Pattern Recognition and Machine Learning.* (Springer New York, 2006).

[9] Wainwright M. J. & Jordan M. I. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn*, **1**, 1-305 (2008).

[10] Tipping M. Sparse bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.* **1**, 211-244 (2001).

[11] Yamashita O. et al. Sparse estimation automatically selects voxels relevant for the decoding of fmri activity patterns, *Neuroimage* **42**, 1414-1429 (2008).

[12] Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proceedings of the 14th international joint conference on Artificial intelligence, 1137-1143 (1995).

[13] Reger B. et al. Connecting brains to robots: an artificial body for studying the computational properties of neural tissues, *Artif. Life* **6**, 307 (2000).

[14] Fang X., Misra S., Xue G. & Yang D. Smart grid the new and improved power grid: A survey. *IEEE Commun. Surv. Tutor.* **14**, 944-980 (2012).

[15] Jabr R. Minimum loss operation of distribution networks with photovoltaic generation. *IET Renew. Power Gener.* **8**, 33-44 (2014).

[16] Zimmerman R., Murillo-Sanchez C. & Thomas R. MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education. *IEEE Trans. Power Syst.* **26**, 12-19 (2011).

[17] Hameed Z., Hong T. & Cho Y. Condition monitoring and fault detection of wind turbines and related algorithms. *Renew. Sust. Energ. Rev.* **13**, 1-39 (2009).

[18] Yuan Y. et al. Artificial intelligent diagnosis and monitoring in manufacturing. *Arxiv*, (2019).

[19] He Q. et al. Gearbox fault diagnosis based on RB-SSD and MCKD. *China Mechanical Engi-*

neering **28**, 1528-1534 (2017).

[20] Lygeros, J. Tomlin, C. & Sastry, S. *Hybrid Systems: Modeling, Analysis and Control* (UC Berkeley / ETH Zurich lecture notes, 2008)

[21] Ljung, L. *System identification: theory for the user* (PTR Prentice Hall, Upper Saddle River, NJ 1999).

[22] Brunton, S. L., Proctor, J. L. & Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl. Acad. Sci. USA* **113**, 3932-3937 (2016).

[23] Bako, L. Identification of switched linear systems via sparse optimization. *Automatica* **47**, 668-677 (2011).

[24] Ly, D. L. & Lipson, H. Learning symbolic representations of hybrid dynamical systems. *J. Mach. Learn. Res.* **13**, 3585-3618 (2012).

[25] Baran, M. & Wu, F. Network reconfiguration in distribution systems for loss reduction and load balancing. *IEEE Trans. Power Deliv.* **4**, 1401-1407 (1989).

[26] Courtemanche, M., Ramirez, R. & Nattel, S. Ionic mechanisms underlying human atrial action potential properties: insights from a mathematical model. *Am. J. Physiol.-Heart Circul. Physiol.* **275**, 301-321 (1998).