Revealing hidden dynamics from time-series data by ODENet

Pipi Hu\textsuperscript{1,*}, Wuyue Yang\textsuperscript{1,*}, Yi Zhu\textsuperscript{1†}, and Liu Hong\textsuperscript{1‡}

\textsuperscript{1}Zhou Pei-Yuan Center for Applied Mathematics, Tsinghua, Beijing, 100084, China.

May 12, 2020

Abstract

To understand the hidden physical concepts from observed data is the most basic but challenging problem in many fields. In this study, we propose a new type of interpretable neural network called the ordinary differential equation network (ODENet) to reveal the hidden dynamics buried in the massive time-series data. Specifically, we construct explicit models presented by ordinary differential equations (ODEs) to describe the observed data without any prior knowledge. In contrast to other previous neural networks which are black boxes for users, the ODENet in this work is an imitation of the difference scheme for ODEs, with each step computed by an ODE solver, and thus is completely understandable. Backpropagation algorithms are used to update the coefficients of a group of orthogonal basis functions, which specify the concrete form of ODEs, under the guidance of loss function with sparsity requirement. From classical Lotka-Volterra equations to chaotic Lorenz equations, the ODENet demonstrates its remarkable capability to deal with time-series data. In the end, we apply the ODENet to real actin aggregation data observed by experimentalists, and it shows an impressive performance as well.

Keywords— time-series data, ordinary differential equations, ODENet, symbolic regression, chemical reactions

1 Introduction

At every moment, massive data has been collected in all fields of human activity. The changes of the data at different time present the evolution of our nature from different aspects. One fundamental goal of science to reveal the hidden dynamics from the time-series data. Thanks to the development of human knowledge, there are many “standard” theories to describe such dynamics, e.g. differential equations are among the most successful ones. However, in many other fields, such as molecular biology, finance, and so on, no such standard theory is available till now. Therefore, modeling the dynamics from the data-driven view is a vital task. Unfortunately, the recorded time-series data usually contain a lot of missing points and even flaws. They are also highly noisy, with useful signals deeply buried. These facts make analyzing time-series data and extracting useful models or principles hard and tricky. A most famous example is the explanation of planetary orbits in the solar system. Even though wrongly placing the earth at the center, Claudius Ptolemy still was able to explain the

\textsuperscript{*}These authors have contributed equally to this work
\textsuperscript{†}yizhu@mail.tsinghua.edu.cn
\textsuperscript{‡}zcamhl@tsinghua.edu.cn
motion of planets and the sun by obscure deferent and epicycle with certain accuracy. Only 1,400 years later, after the landmark work of Copernicus, Kepler, and Newton, a much simpler and more correct theory could be gradually established and accepted. This story highlights the ambiguity and difficulty behind data-driven modeling. Influenced by the great success of modern machine learning algorithms, we aim to reveal the hidden dynamics purely from the time-series data without prior knowledge. Different from most previous works which focus on the efficiency of predictions of neural networks without any understandings, this work applies a powerful tool to construct the proper governing evolutionary equations which is suitable to describe the dynamics.

In the recent years, analysis of time-series data has already become a specific subject in modern science [1]. Especially in the presence of big data, plenty of machine learning algorithms, like recurrent neural network (RNN) [2], Long short-term memory (LSTM) [3], etc., have been widely applied to this subject. RNN, which uses its internal state network to store representations of recent inputs and reuse the output to process the time series, shows a dramatic different “memory” property and network flow structure from other supervised neural networks. The RNN plays important roles in natural language processing (NLP), non-Markovian control and etc. LSTM solves the problems of vanishing gradients in RNN by using the so-called “long term” and “short term” memories. Deep learning can extract much higher level features progressively from the input and is thus suitable for the analysis of time-series data. Though RNN and LSTM are very successful in many fields, their performance in dealing with time-series date collected from physical processes are not good [4]. By adding short-cuts to jump over some layers, the ResNet [5] can avoid the problem of vanishing gradients and shows better performances than classical deep learning networks. Those short-cuts can also be treated as some kind of “memory”, which keeps the intrinsic properties unchanged during the learning procedure. The simplest structure of ResNet in a mathematical form is

\[ x_{n+1} = f(x_n; \theta) + x_n \]

where \( x_n \) is the input of \( n \)th network or the output of \( (n-1) \)th network, so is \( x_{n+1} \), and \( f(x_n) \) is the \( n \)th network mapping function with parameters \( \theta \). After simple rearrangement, Eq. (1) can be rewritten as

\[ \frac{x_{n+1} - x_n}{h} = \frac{1}{h} f(x_n; \theta), \]

which is the Euler’s scheme of ordinary differential equations (ODEs)

\[ \frac{dx}{dt} = \tilde{f}, \quad \text{where} \quad \tilde{f} = \frac{1}{h} f(x; \theta). \]

From this point of view, the ResNet takes full advantage of the “memory”, since it mimics the solution of an ordinary differential equations over the whole time region. We see that from RNN to ResNet, more and more “memories” are included.

The mathematical analogy between ResNet and numerical schemes of ODEs is of great significance, which actually provides a direct linkage between the procedure of learning a time series and extracting the hidden dynamics from the data. This continuous view of machine learning has been studied both in the mathematical analysis by Weinan E et al. [6, 7] and the algorithm construction of machine learning by Maziar Raissi et al. [8] and Chen et al [4]. On the other hand, extracting dynamics from time-series data attracts enormous attentions these years. For example, Bongard and Schmidt used symbolic regression to find nonlinear differential equations [9, 10, 11]. Kutz et al. proposed to use sparse identification for nonlinear dynamical systems [12]. In order to stabilize the performance of sparse identification in the presence of noise, some technical and empirical schemes for differential operations were proposed [13, 14].

In this study, we are going to address a new type of neural network, called the ordinary differential equation network (ODENet), and its performance on extracting unknown governing dynamical equations from pre-given time-series data. By combining ODENet with symbolic regression, sparse identification, and signal-noise decomposition, we are able to obtain explicit ODE models in high
accuracy for population dynamics modeled by Lotka-Volterra equations, strange-attractors of Lorenz
equations in the chaotic region, and actin aggregation dynamics under distinct experimental condi-
tions and molecular mechanisms. Furthermore, our framework is robust and noise-tolerant, which
therefore is quite suitable for time-series data analysis and data-driven mathematical modeling.

2 The architecture of ODENet

Basically, there are two ways to interpret the time series data. The usual machine learning algo-
rithms, like LSTM and deep learning, tend to use a vast neural network containing a large number
of free parameters to achieve the goal of best data fitting at each point. Through iterative training
and optimization, the data correlation is transformed into very complicated and thus unexplainable
relations among network nodes. In contrast, regression and sparse identification methods adopt
an alternative view, which more concerns about the construction of explicit relations or dynamic
equations for a globally fitting of all data at the same time but only with a few parameters. Each
way has its own advantages and appropriate applicable regions. Since we are more interested in the
physical mechanisms and mathematical models behind the data, the two ways are combined and
realized through the ODENet in the current study.

As a modification of ResNet, the basic structure of ODENet, as shown in Figure 1, is mimicking
the numerical solvation of ODEs with unspecified parameters. Through iteratively minimizing the
difference between predicted values and training data measured by certain loss function, unknown
parameters are optimized in such a way that the most suitable ODE model for the given data is
explicitly specified.

With respect to pre-given time series data, the learning procedure of an ODENet starts with
randomly selected \( n + 1 \) successive time points \( t_0 < t_1 < t_2 < \cdots < t_n \) and their corresponding values
\( x(t_0), x(t_1), x(t_2), \cdots, x(t_n) \), where \( x(t) = (x_1, x_2, \cdots, x_d)^T \) is a \( d \)-dimension vector representing data
values at time \( t \). Next, we solve the initial value problem of an autonomous equation
\[
\frac{dx}{dt} = f(x; \theta),
\]
\[
x(t_0) = (x_1(t_0), x_2(t_0), \cdots, x_d(t_0))^T,
\]
by an ODE solver (e.g. the Runge-Kutta method). Note \( x(t_0) \) at \( t_0 \) is taken as the initial value here.

In Eq. (3), \( f(x; \theta) = (f_1, f_2, \cdots, f_d)^T \) are linear or nonlinear functions whose explicit forms are
specified by basis functions with corresponding coefficients \( \theta \). Although the selection of proper basis
function is quite tricky and problem dependent, polynomials are among the most often used ones in
practice. For the \( d \)-dimensional vector \( x = (x_1, x_2, \cdots, x_d)^T \), the \( p \)-th order complete polynomials
\( \{1, x_1, x_2, \cdots, x_d, x_1 x_1, x_1 x_2, \cdots, x_1^p \} \) have \( M = \binom{p+d}{p} - 1 \) terms. So that we have
\[
f(x; \theta) = \theta \Lambda,
\]
where \( \Lambda = (1, x_1, x_2, \cdots, x_d, x_1 x_1, x_1 x_2, \cdots, x_1^p)^T \) is the complete set of \( p \)-th order polynomial basis
with coefficients \( \theta = (\theta_{ij})_{d \times M} \).

The adoption of ODE solver than an ordinary neural network like ResNet offers a dramatic
advantage in dealing with input data at non-equally spaced time points directly and efficiently.
However, a price we have to pay is the stiffness problem, especially when linear combinations of
high-order polynomials are taken as the right-hand side of an ODE system. So as a general rule, we
suggest using self-adaptive ODE solvers and implicit ODE solvers, which are designed for solving
stiff cases for the best [15, 16].
Figure 1: The structure of ODENet with key procedure highlighted in the right columns.
The outputs of an ODE solver are predicted values of $x(t)$ at time $t_1, t_2, \ldots, t_n$, which are denoted as $\tilde{x}(t_1), \tilde{x}(t_2), \ldots, \tilde{x}(t_n)$ respectively. Then by tuning free parameters $\theta$, which specify the concrete form of ODEs, the loss function

$$L(\theta; x) = \|x - \tilde{x}\|_2 + \alpha \|\theta\|_1$$

is expected to be minimized. Here the first term characterizes the difference between training data and predictions, while the second term represents the sparsity requirement with $\alpha$ as a hyper parameter. Due to Occam’s Razor, the derived ODE model should be as simple as possible, so it is expected that most components in $\theta$ would be zeros, which thus leads to a small L1 norm.

The searching of best $\theta$ is a global optimization problem, which can be done via mature learning algorithms in the neural network, for example, gradient descent or stochastic gradient descent combined with backward propagation algorithms (BP)[17]. Especially in Pytorch, this can be simply done by an integrated autograd library. Once the forward flow is constructed, the backward flow will be automatically built by Pytorch [18]. Finally, repeating the above procedure iteratively until the loss function does not decay efficiently anymore or less than a threshold, we finish the learning procedure of ODENet, which is summarized through pseudocodes as below.

**Algorithm 1: Pseudocode of ODENet (Pytorch)**

```
Input: time-series data $x(0), x(t_1), \ldots, x(t_n)$, where $x = (x_1, x_2, \ldots, x_d)^T$.
Output: $d$-dimensional first order ODEs $\frac{dx}{dt} = f(x, \theta)$.
initialize parameters $\theta$;
initialize hyperparameters: threshold_L, threshold_\theta, m,n;
while $L >$ threshold_L do

// Construct one batch
select $n$-length intervals from $m$ random positions for a batch;
batch ← $[x_1^1, x_1^2, \ldots, x_1^n], [x_2^1, x_2^2, \ldots, x_2^n], \ldots, [x_m^1, x_m^2, \ldots, x_m^n]$;
batch_f ← $[t_1^1, t_1^2, \ldots, t_1^n], [t_2^1, t_2^2, \ldots, t_2^n], \ldots, [t_m^1, t_m^2, \ldots, t_m^n]$;
batch_label ← batch[:1];
batch_size ← $m$;
L ← 0;
for $i ← 1$ to batch_size do

// Compute the predictions based on batch_init
pred[i,:] ← ODESolve(batch_init[i], batch_f[i,:], \theta);
// The \theta matrix should be sparse matrix
$L ← L + \|\text{Abs}(\text{pred[i,:]} - \text{batch_label[i,:]}\|_2 + \epsilon\|\theta\|_1$;
end
\theta_grad ← \frac{\partial L}{\partial \theta}$ by BP algorithm;
Update parameters $\theta$ by Adam methods;
// Set \theta_{ij} which is small enough to be zero\n$\theta[\theta < \text{threshold}_\theta] ← 0$;
end
```

As we have discussed in the introduction, real data can not be perfect. Noise and faults are seen from place to place. To deal with this issue, we treat noise as learning parameters too. Suppose there is a finite time series $y(t) \in \mathbb{R}^d$ with noise $e(t) = \epsilon \|y\|_\infty \eta$, where $\epsilon$ denotes the noise strength, $\|y\|_\infty$ is the maximal value in $y$, and $\eta \sim \mathcal{N}(0, 1)$ are $d$-dimensional normally distributed random
variables. ODENet is proposed to extract the following autonomous system of ODEs

$$\frac{dx}{dt} = f(x; \theta), \ x(t) = y(t) - e(t) \in \mathbb{R}^d. \quad (6)$$

This time, the loss function depends on $e(t)$ too, i.e. $L = L(\theta, e(t); \alpha)$. And we can apply the same process as before to extract the governing dynamics from the data series.

### 3 Numerical experiments

In this section, we are going to apply the ODENet to the study of Lotka-Volterra equations in diverse parameter regimes and Lorentz equations in the chaotic regime. Through these examples, the power and advantage of ODENet could be learned clearly.

#### 3.1 Lotka-Volterra equations with and without large noise

**Goal:** “Find” the correct Lotka-Volterra (LV) model from the given time-series data.

**Data:** Simulated time trajectories of LV equations representing different types of ODE dynamics in phase space, combined with either small (1%) or large (10%) white noises.

**Setup:** Complete polynomials up to the second order $\Lambda = \{1, x_1, x_2, x_1^2, x_1x_2, x_2^2\}$ with twelve adjustable coefficients $\theta = (\theta_{ij})_{2 \times 6}$ are adopted to approximate function $f(x)$. For large noise, the noise term $e(t)$ in (6) is added as learning parameters, too.

**Learning:** Optimize parameters $\theta$ and $e$ following the procedure of Algorithm 1. Parameters $\theta_{ij}$ less than the threshold and unaffecting the dynamics dramatically is set as zero to remove model redundancy.

**Results:** The correct form of LV models is reproduced with most terms as zeros. The coefficients of remaining terms are close to their expected values, with the maximal relative errors less than 6%. The distribution of noise is almost correctly predicted.

The Lotka-Volterra (LV) equations, also known as predator-prey equations, were first introduced by Lotka[19] and Volterra[20] in the 1920s to describe the population dynamics of preys interacting with predators in ecological systems. LV equations have been widely applied to ecological balance[21], environmental protection[22], disease prevention and control[23], etc. A very general form of LV equations including species growth and death, intraspecies and interspecies competition reads

$$\begin{align*}
\frac{dx_1}{dt} &= C_{11}x_1 + C_{12}x_1x_2 + C_{13}x_1^2, \\
\frac{dx_2}{dt} &= C_{21}x_2 + C_{22}x_1x_2 + C_{23}x_2^2. \\
\end{align*} \quad (7)$$

It’s easy to see above equations have four fixed points, i.e. $(0, 0), \left(-\frac{C_{13}}{C_{12}}, 0\right), \left(0, -\frac{C_{21}}{C_{22}}\right)$ and \left(-\frac{C_{12}C_{23} - C_{11}C_{22}}{C_{12}C_{22} - C_{11}C_{23}}, \frac{C_{13}x_1}{C_{12}x_2} \right) \text{ under the condition } C_{12} \neq C_{13} \neq C_{21} \neq C_{22} \neq C_{23} \neq C_{23} \neq 0. \text{ And the dynamic behaviors of LV equations around these fixed points } (x_1^*, x_2^*) \text{ are fully specified by the Jacobian matrix (its eigenvalues to be exact)}

$$J = \begin{bmatrix} C_{11} + C_{12}x_2 + 2C_{13}x_1 & C_{12}x_1 \\ C_{22}x_2 & C_{21} + C_{22}x_1 + 2C_{23}x_2 \end{bmatrix}_{(x_1^*, x_2^*)}. \quad (8)$$
which can be roughly classified into three basic types – the extinction of one species, or the evolution to an equilibrated coexistence, or to a continuing oscillation (limit cycle) [24].

| cases         | Trajectories | Phase portrait |
|---------------|--------------|----------------|
| (21) over damped | ![trajectory](image1.png) | ![phase portrait](image2.png) |
| (22) spiral    | ![trajectory](image3.png) | ![phase portrait](image4.png) |
| (23) limit cycle | ![trajectory](image5.png) | ![phase portrait](image6.png) |

Figure 2: The accuracy of ODENet predictions in both time region (left column) and phase space (right column) in comparison with exact solutions of LV equations in the presence of 1% white noises.

Based on the above analysis, we implement the ODENet to learn the dynamics of pre-given LV
equations in different coefficient regimes, as shown in Figure 2. At first, we do not include the influence of noise. And it can be clearly seen that the ODENet takes the best advantage of the input time-series data. The predicted six coefficients in LV equations are so close to the real ones with the relative errors less than 6% (see Table 1).

Even in the presence of large noises, for example in this case up to 10% white noise with respect to the maximal signal value are added to the data (see Fig. 3), our ODENet still shows an astonishing ability in finding out the correct governing equations and revealing the hidden deterministic trajectories which are deeply buried inside noise-spoiled data. The learned noises correctly fit into a Gaussian distribution as expected, though rare events with large displacements are overestimated in the current case, as pointed out in Fig. 3c. Most importantly, in ODENet, no extra unneeded coefficient will be included in the model as a consequence of sparse identification, even for the flawed and noisy data. This fact is clearly stated through the zero values of $C_{13}$ and $C_{23}$ in the fourth row of Table 1 for LV equations with white noise.

### 3.2 Lorenz equations in chaotic regimes

In 1890, Poincaré made a landmark finding that the three-body system is unstable. A small disturbance in the initial state would make the future state completely unpredictable. Poincaré’s work put forward a great challenge on the traditional view of determinism and draw people’s attention to chaos for the first time. Later, in the 1960s, American meteorologist Lorenz proposed a simple mathematical system constituted by three ordinary differential equations,

\[
\begin{align*}
\frac{dx_1}{dt} &= C_{11}x_1 + C_{12}x_2, \\
\frac{dx_2}{dt} &= C_{21}x_1 + C_{22}x_2 + C_{23}x_1x_3, \\
\frac{dx_3}{dt} &= C_{31}x_3 + C_{32}x_1x_2.
\end{align*}
\]  

(9)

for describing atmospheric turbulence [25]. Lorenz equations becomes very famous for its chaotic solutions. For a typical parameter combination $C_{11} = -C_{12} = \sigma = 10, C_{21} = r = 28, -C_{32} = b = 8/3, -C_{22} = -C_{23} = C_{31} = 1$, the Lorenz system has three equilibrium points, i.e., $(0,0,0)$,
Figure 3: (a-b) Predictions of ODENet on LV equations in the presence of large external noises (up to 10% of the signal value). Distributions of learned noises in (c) $x_1$ and (d) $x_2$ were compared with pre-given ones.
(6√2, 6√2, 27) and (−6√2, −6√2, 27). Numerical simulation shows that typical trajectories of Eq. 9 follow a strange attractor in a butterfly shape in the phase space, which first makes a few loops around (6√2, 6√2, 27), then jumps to loops around (−6√2, −6√2, 27), and then come back to the point (6√2, 6√2, 27), again and again (see Figure 4). In this case, solutions of the Lorenz equations are sensitive to the disturbance in the initial conditions, which is widely known as the “butterfly effect” in the literature. Therefore, to catch the charming butterfly from pre-given chaotic data becomes an attractive task and a benchmark problem in testing the accuracy of numerical schemes as well as the performance of machine learning algorithms.

To raise up the learning difficulty, we introduced white noises with values up to 0.5% of the maximal signal data, though no disturbance is included in the initial values due to the butterfly effect. According to the results summarized in Table 2 and Figure 4, ODENet correctly reproduces the Lorenz attractor, and only small fraction trajectories are mispredicted, which is inevitable in the study of chaos.

### Box 2: Strange attractors of Lorentz equations

**Goal:** Predict the strange attractors of Lorentz equations.

**Data:** Time trajectories of Lorenz equations in the chaotic regime with 0.5% white noise added.

**Setup:** Complete polynomials up to the second order \( \Lambda = \{1, x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3\} \) with thirty adjustable coefficients \( \theta = (\theta_{ij})_{3 \times 6} \) are adopted to approximate function \( f(x) \). As the orbits are very sensitive to initial values and model coefficients, they are divided into many small intervals to minimize predictive errors.

**Learning:** Parameters \( \theta \) are optimized according to the procedure of Algorithm 1. Sparsity requirement is taken.

**Results:** The correct form of Lorenz equations is reproduced with the maximal relative errors of coefficients less than 1%. The strange attractors are basically correctly predicted even in the long time.

| Lorentz Parameters | C_{11} | C_{12} | C_{21} | C_{22} | C_{23} | C_{31} | C_{32} |
|--------------------|--------|--------|--------|--------|--------|--------|--------|
| true value         | -10    | 10     | 28     | -1     | -1     | -8/3   | 1      |
| prediction         | -9.989 | 9.982  | 28.02  | -1.008 | -1.000 | -2.667 | 1.000  |
| relative err       | -0.1%  | -0.2%  | 0.1%   | 0.8%   | 0.0%   | 0.0%   | 0.0%   |

Table 2: The learned parameters for Lorenz equations.

### 4 Application to kinetics of actin aggregation

Actin aggregation into microfilaments is responsible for the contraction of muscle cells and the motility of other cells. In the 1960s, the first analytical molecular model was proposed by Oosawa et al.[26], which stated the mechanism of actin aggregation includes three basic steps — primary nucleation, elongation and fragmentation. Primary nucleation is a pre-step to generate new growth
Figure 4: Predictions of ODENet on Lorentz equations in the chaotic regime.
seeds through a self-organization process. Then small seeds grow into long actin filaments by elongation, meaning monomeric actins are added to the filament ends in a sequential way. The actin aggregation could be dramatically speeded up by fragmentation, through which massive new seeds are generated by breaking long filaments into two smaller pieces without involving primary nucleation. It should be mentioned, besides those forward processes for actin growth, the corresponding inverse processes, like monomer dissociation and fibril annealing may also make a non-negligible contribution to maintaining the equilibrium distribution of actin filaments. Based on the theory of chemical kinetics, the above picture can be explicitly transformed into a mathematical language of ordinary differential equations, which establish a direct connection between experimental data and molecular mechanisms of actin growth.

In this study, we reexamine the classical experiments done by Wegner et al. [27], which studied the phenomenon of actin aggregation under two distinct conditions. One is varied concentrations of monomeric actins from 7.4µM to 20.5µM incubated with 40mM KCl (Figure. 5a), the other is 6.7 – 22.9µM actins incubated with 0.6mM MgCl2 and 0.5 mM EGTA (Figure. 5b). To explore the influence of pre-knowledge (or physical insight) on machine learning-based modeling, here we adopt two different setups – one is purely data-driven, the other is physical-based, which as we will see, leads to models in distinct forms, but all fit the data quite well.

**Box 3: Data-driven v.s. physical-based modeling of actin aggregation**

**Goal:** Compare the purely data-driven model with the physical-based model on kinetics of actin aggregation.

**Data:** Mass concentration $M(t)$ of actin filaments recorded at different time points in ThT fluorescence experiments. Seven protein concentrations and two buffer conditions are taken into consideration.

**Setup:** There are two separate setups for the learning procedure:

1. **Purely data-driven model.** Without any pre-knowledge of the model, we just need single equation of mass concentration $M(t)$ to learn the dynamics. To account for the concentration dependence, an additional variable $m(t) = m_{tot} - M(t)$ is introduced too. Mimicking the function on the right-hand side of ODEs by polynomials up to the second-order, we have to optimize six coefficients $\theta = (\theta_{ij})_{2 \times 6}$, where $\theta_{2j} = -\theta_{1j}$, $j = 1, 2, \ldots, 6$.

2. **Physical based model.** According to the general theory for actin aggregation [28], another hidden variable – the number concentration of actin filaments $P(t)$ is introduced into the model. Furthermore, we require all kept terms have a clear physical meaning to account for all possible mechanisms for actin growth. In this case, we have three ordinary differential equations with eleven undetermined coefficients. The hidden variable $P(t)$ is constructed in a self-iterative way from the approximation $dM/dt \approx \alpha g m P$ with a pre-knowledge that elongation makes a major contribution to the mass growth of actin filaments.

**Learning:** Parameters $\theta$ are optimized according to the procedure of Algorithm 1. Sparsity requirement is taken.

**Results:** Two simple models with and without hidden physical variable $P(t)$ are learned separately, both of which can fit ThT trajectories quite well.

Firstly, we study the pure data-driven modeling without including any pre-knowledge. Besides
For simplicity, terms on the right-hand side are kept up to the second-order polynomials of $M$ intensity, an additional variable – the actin monomer concentration $m$ – the mass concentration of actin filaments $M$ the mass concentration of actin comes from actin filament elongation, which depends on not only the monomer concentration but also the filament concentration. Only the physical meaning of negative as expected. The term $\alpha_3$ represents the process of degradation (or monomer dissociation). Since it makes a negative contribution to the filament concentration, $\alpha_1$ is always negative as expected. The term $\alpha_4 M$ comes from actin filament elongation, which depends on not only the monomer concentration but also the filament concentration. Only the physical meaning of $\alpha_3 M^2$ is not so straightforward, which may originate from some complicated interactions between filaments, like annealing or clumping. However, based on the coefficients listed in Table 3, we cannot clearly tell the difference between actin incubating with KCl and with MgCl$_2$. These limitations motivate us to consider a more physical-based model.

Leaving alone the good agreement between ODENet predictions and experimental data as shown in Figure 5, the learned ODE parameters for actin aggregation given in Table 3 are worthy of further clarification, especially their physical meanings. Terms $\alpha_0$, $\alpha_2 m$ and $\alpha_5 m^2$ together account for primary nucleation within two monomers. $\alpha_1 M$ represents the process of degradation (or monomer dissociation). Since it makes a negative contribution to the filament concentration, $\alpha_1$ is always negative as expected. The term $\alpha_4 M$ comes from actin filament elongation, which depends on not only the monomer concentration but also the filament concentration. Only the physical meaning of $\alpha_3 M^2$ is not so straightforward, which may originate from some complicated interactions between filaments, like annealing or clumping. However, based on the coefficients listed in Table 3, we cannot clearly tell the difference between actin incubating with KCl and with MgCl$_2$. These limitations motivate us to consider a more physical-based model.

According to the general theory for actin aggregation [28], besides the mass concentrations of actin filaments and monomers, the number concentration of actin filaments $P$ also plays a non-negligible role in constructing a complete description of actin growth. So instead of two ODEs, in principle we should consider three coupled equations as the “correct” model. However, as the experimental data contain no direct information on $P$, the variable $P$ is actually a hidden one. If we do not write it out explicitly, there is no way to learn it in a purely data-driven modeling. Up to the second-order polynomials, we have

$$\begin{align*}
\frac{dM}{dt} &= \alpha_0 + \alpha_1 M + \alpha_2 m + \alpha_3 M^2 + \alpha_4 mM + \alpha_5 m^2, \\
\frac{dm}{dt} &= -\alpha_0 - \alpha_1 M - \alpha_2 m - \alpha_3 M^2 - \alpha_4 mM - \alpha_5 m^2.
\end{align*}$$

(10)

For simplicity, terms on the right-hand side are kept up to the second-order polynomials of $M$ and $m$. Due to the laws of mass conservation, i.e. $M(t) + m(t) = m_{tot}$, five free parameters in the second equation of $m$ can be removed. It is further noted that, since in the current case at least seven concentrations of actin are considered, a global fitting of data with different $m_{tot}$ at the same time is essential for the learning procedure of ODENet.

Table 3: Learned parameters for data-driven model of actin aggregation.

| filaments       | Parameters                           |
|-----------------|--------------------------------------|
| actin in KCl    | $\alpha_0$ 4.62×10^{-1} $\alpha_1$ -2.16×10^{-1} $\alpha_2$ -5.49×10^{-1} $\alpha_3$ 5.70×10^{-3} $\alpha_4$ 1.10×10^{-1} $\alpha_5$ 7.87×10^{1} |
| actin in MgCl$_2$ | 0 $\alpha_1$ -1.41×10^{-2} 9.20×10^{-3} -3.75×10^{-2} 2.28×10^{-1} 3.10×10^{1} |

Here those terms without any physical meaning have been removed, and only eleven free coeffi-

13
Figure 5: Kinetics of actin aggregation in (a) KCl and (b) MgCl₂ solutions respectively. Red circles for experimental data in [27], blue solid lines for predictions of the data-driven model in (10), green dashed lines for the physical-based model in (11).

cients are kept instead of thirty. The remaining terms all have clear physical interpretations. To be exact, terms \( \alpha_0 \), \( \alpha_1m \) and \( \alpha_2m^2 \) stand for primary nucleation, \( \alpha_3mM \) for secondary nucleation, \( \alpha_4P \) and \( \alpha_5M \) for fibril degradation, \( \alpha_6M \) for fragmentation, \( \alpha_7P^2 \) for annealing, \( \alpha_7PM \) for clumping, \( \alpha_9mP \) for elongation and \( \alpha_{10}P \) for monomer dissociation respectively.

During the learning procedure of ODENet, terms \( \alpha_0 \), \( \alpha_5M \), \( \alpha_6P^2 \), \( \alpha_7PM \) and \( \alpha_8M \) are eliminated by sparsity requirement, indicating the corresponding processes may not be essential for modeling. The remaining terms listed in Table 4 suggest a clear molecular mechanism for the actin aggregation, including primary nucleation (indicated by \( \alpha_1m \) and \( \alpha_2m^2 \)), elongation (\( \alpha_9mP \)), surface catalyzed secondary nucleation (\( \alpha_3mM \)), monomer dissociation (\( \alpha_{10}P \)) and degradation \( \alpha_4P \). Among them, the first three processes are dominant for microfilament growth, while the latter two are responsible for maintaining the equilibrium state. Further comparing the model coefficients learned from ODENet, the elongation rate for actin aggregation in KCl solution is much smaller than in MgCl₂ solution, suggesting the former process is dominated by primary nucleation, while the latter is dominated by elongation and secondary nucleation instead. This dramatic distinction is believed to be caused by different chemical valences of K⁺ and Mg²⁺. Therefore, the physical based modeling by ODENet indeed provides new insights into those unknown phenomena we are interested in.

5 Conclusion

Nowadays, with the rapid advances in high-through instruments, massive data have been accumulated in physics, chemistry and biology. However, most of the time, how to extract useful patterns from the data is still a quite challenging task, let alone the hidden dynamics and even mechanisms. Luckily, machine learning algorithms, either unsupervised or supervised, provide an automatic and reliable way to achieve this goal. In this study, we present how to use a new type of interpretable...
Table 4: Learned parameters for physical-based model of actin growth. Unmentioned parameters are all zero.

| parameters | $\alpha_1$ | $\alpha_2$ | $\alpha_3$ | $\alpha_4$ | $\alpha_9$ | $\alpha_{10}$ |
|------------|------------|------------|------------|------------|------------|------------|
| actin in KCl | $-5.12 \times 10^{-2}$ | $7.98 \times 10^{-3}$ | $1.16 \times 10^{-2}$ | $-5.33 \times 10^{-1}$ | $7.39 \times 10^{-1}$ | $-8.82 \times 10^{-1}$ |
| actin in MgCl$_2$ | $2.15 \times 10^{-2}$ | $0$ | $2.3 \times 10^{-2}$ | $0$ | $1.19 \times 10^{1}$ | $-2.97 \times 10^{1}$ |

neural network, called ODENet to deal with time-series data modeled by ordinary differential equations. This kind of data and models are frequently met in chemical reactions, population dynamics, classical mechanics and so on, once the system is homogeneous in space or well-mixed. As illustrated through examples including Lotka-Volterra models for population dynamics, strange attractors of Lorenz equations, kinetics of actin aggregation into microfilaments, ODENet shows great advantages in several aspects: (1) the ability to deal with data not equally spaced, of a low signal to noise ratio, etc.; (2) explicitly deriving understandable models with fewer parameters; (3) efficient optimization of parameters by BP algorithms; (4) very flexible network structure ready for the incorporation of various new approaches; (5) quick convergence rate and high learning speed. Therefore, we expect much wider applications of ODENet in various branches of natural science, as well as nontrivial extensions to stochastic ODEs and PDEs for a better description of data in the real world.

6 Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant Nos. 21877070 and 11871299) and the 13th 5-Year Basic Research Program of CNPC (2018A-3306).

References

[1] William WS Wei. Time series analysis. In The Oxford Handbook of Quantitative Methods in Psychology: Vol. 2. 2006.

[2] Jeffrey L. Elman. Finding structure in time. Cognitive science, 14(2):179–211, 1990.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[4] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In Advances in neural information processing systems, pages 6571–6583, 2018.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[6] E Weinan. A proposal on machine learning via dynamical systems. Communications in Mathematics and Statistics, 5(1):1–11, 2017.
[7] Chao Ma, Lei Wu, et al. Machine learning from a continuous viewpoint. *arXiv preprint arXiv:1912.12777*, 2019.

[8] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236*, 2018.

[9] Josh Bongard and Hod Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, 2007.

[10] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[11] John R Koza and John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[12] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937

[13] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017.

[14] Samuel H Rudy, J Nathan Kutz, and Steven L Brunton. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *Journal of Computational Physics*, 396:483–506, 2019.

[15] M Calvo, JI Montijano, and L Randez. A fifth-order interpolant for the dormand and prince runge-kutta method. *Journal of computational and applied mathematics*, 29(1):91–100, 1990.

[16] Ernst Hairer and Gerhard Wanner. Stiff differential equations solved by radau methods. *Journal of Computational and Applied Mathematics*, 111(1-2):93–111, 1999.

[17] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[18] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[19] Alfred J Lotka. Elements of mathematical biology. 1956.

[20] P Volterra. Lessons in the mathematical theory of the struggle for life, 1931.

[21] Paul A Samuelson. Generalized predator-prey oscillations in ecological and economic equilibrium. *Proceedings of the National Academy of Sciences*, 68(5):980–983, 1971.

[22] Bi-Huei Tsai, Chih-Jen Chang, and Chun-Hsien Chang. Elucidating the consumption and co2 emissions of fossil fuels and low-carbon energy in the united states using lotka–volterra models. *Energy*, 100:416–424, 2016.

[23] Robert D Holt and John Pickering. Infectious disease and species coexistence: a model of lotka-volterra form. *The American Naturalist*, 126(2):196–211, 1985.
[24] Robert M May. Limit cycles in predator-prey communities. *Science*, 177(4052):900–902, 1972.

[25] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141, 1963.

[26] Fumio Oosawa and Michiki Kasai. A theory of linear and helical aggregations of macromolecules. *Journal of molecular biology*, 4(1):10–21, 1962.

[27] Albrecht Wegner and Paula Savko. Fragmentation of actin filaments. *Biochemistry*, 21(8):1909–1913, 1982.

[28] Liu Hong, Chiu Fan Lee, and Ya Jing Huang. Statistical mechanics and kinetics of amyloid fibrillation. 9, 2017.