Data-driven discovery of multiscale chemical reactions governed by the law of mass action

Juntao Huang [1]   Yizhou Zhou [2]   Wen-An Yong [3]

Abstract

In this paper, we propose a data-driven method to discover multiscale chemical reactions governed by the law of mass action. First, we use a single matrix to represent the stoichiometric coefficients for both the reactants and products in a system without catalysis reactions. The negative entries in the matrix denote the stoichiometric coefficients for the reactants and the positive ones for the products. Second, we find that the conventional optimization methods usually get stuck in the local minima and could not find the true solution in learning the multiscale chemical reactions. To overcome this difficulty, we propose a partial-parameters-freezing (PPF) technique to progressively determine the network parameters by using the fact that the stoichiometric coefficients are integers. With such a technique, the dimension of the searching space is gradually reduced in the training process and the global mimina can be eventually obtained. Several numerical experiments including the classical Michaelis–Menten kinetics, the hydrogen oxidation reactions and the simplified GRI-3.0 mechanism verify the good performance of our algorithm in learning the multiscale chemical reactions. The code is available at `https://github.com/JuntaoHuang/multiscale-chemical-reaction`.

Key Words: Chemical Reactions; Multiscale; Machine Learning; Nonlinear Regression; Ordinary Differential Equations

[1]Department of Mathematics, Michigan State University, East Lansing, MI 48824, USA. E-mail: huangj75@msu.edu
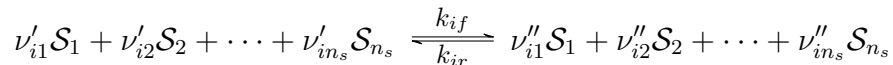
[2]School of Mathematical Sciences, Peking University, Beijing, China. E-mail: zhouyz@math.pku.edu.cn

[3]Department of Mathematical Sciences, Tsinghua University, Beijing, China. E-mail: wayong@tsinghua.edu.cn.

1 Introduction

Chemical reactions are fundamental in many scientific fields including biology, material science, chemical engineering and so on. To identity the reactions from experimental data, the traditional methods are mainly based on some empirical laws and expert knowledge [10]. Recently, thanks to the rapid development of machine learning [19] and data-driven modeling [34, 22, 6, 3, 29, 20, 30, 14], it is desirable to develop a data-driven method of discovering the underlying chemical reactions from massive data automatically.

Consider a reaction system with n_s species participating in n_r reactions:

$$\nu'_{i1}\mathcal{S}_1 + \nu'_{i2}\mathcal{S}_2 + \cdots + \nu'_{in_s}\mathcal{S}_{n_s} \underset{k_{ir}}{\overset{k_{if}}{\rightleftharpoons}} \nu''_{i1}\mathcal{S}_1 + \nu''_{i2}\mathcal{S}_2 + \cdots + \nu''_{in_s}\mathcal{S}_{n_s}$$

for $i = 1, 2, \cdots, n_r$. Here $\mathcal{S}_k$ is the chemical symbol for the k-th species, the nonnegative integers ν'_{ik} and ν''_{ik} are the stoichiometric coefficients of the k-th species in the i-th reaction, and k_{if} and k_{ir} are the direct and reverse reaction rates of the i-th reaction. The reaction is reversible if both k_{if} and k_{if} are positive. Strictly speaking, all elementary chemical reactions are reversible due to microscopic reversibility. However, in real applications, some of the rate constants are negligible, thus the corresponding reactions can be omitted and the retained ones can be considered as irreversible.

Denote by $u_k = u_k(t)$ the concentration of the k-th species at time t for $k = 1, 2, \cdots, n_s$. According to the law of mass action [38], the evolution of u_k obeys the ordinary differential equations (ODEs) [27]

$$\frac{du_k}{dt} = \sum_{i=1}^{n_r}(\nu''_{ik} - \nu'_{ik})\left(k_{if}\prod_{j=1}^{n_s}u_j^{\nu'_{ij}} - k_{ir}\prod_{j=1}^{n_s}u_j^{\nu''_{ij}}\right), \tag{1.1}$$

for $k = 1, 2, \cdots, n_s$. Given the concentration time series data $\{u_k(t_n),\ k = 1, \cdots, n_s,\ n = 1, \cdots, N\}$, our goal is to learn the stoichiometric coefficients ν'_{ik}, ν''_{ik} and reaction rates k_{if} and k_{ir}.

In the literature there are already some works on this topic. In [5], the authors applied linear regressions to infer the chemical reactions, with the assumption that the reactions are at most the result of bimolecular collisions and the total reaction order is not greater than two. In [40], the linear regression was utilized with an L1 objective, which transforms the problem into a mixed-integer linear programming (MILP). This approach suffers from the same restrictive assumptions as in [5]. In [18], the authors presented an approach to infer the stoichiometric subspace of a chemical reaction network from steady-state concentration data profiles, which is then cast as a series of MILP. In [25], some chemically reasonable requirements were considered such as the mass conservation and the principle of detailed balance. The deep neural networks (DNNs) were applied to extract the chemical reaction rate information in [31, 32], but the weights are difficult to interpret physically. In [13], the authors adapted the sparse identification of nonlinear dynamics (SINDy) method [4, 9] to the present problem. However, the approach relies on expert knowledge, which precludes the application in a new reaction system with unknown reaction pathways. Within the framework of SINDy, other works are [2, 1, 24]. In order to improve the performance of SINDy, two additional steps including least-squares regression and stepwise regression in the identification were introduced in [2], which are based on the traditional statistical methods. In [1], SINDy was combined with the DNNs to adaptively model and control the process dynamics. An implicit-SINDy was proposed and applied to infer the Michaelis-Menten enzyme kinetics in [24]. Additionally, a statistical learning framework was proposed based on group-sparse regression which leverage prior knowledge from physical principles in [23]. For example, the mass conservation is enforced in the JAK-STAT reaction pathway for signal transduction in [23].

Our work is mainly motivated by [15], where the authors proposed a Chemical Reaction Neural Network (CRNN) by resorting to the feature of the equations in (1.1). The discovery of chemical reactions usually involves two steps: the identification of the reaction pathways (i.e., the stoichiometric coefficients) and the determination of the reaction rates. For complex

reaction processes, one could not even identify the reaction pathways and has to infer both the stoichiometric coefficients and the rate constants from data. The work in [15] presents a neural network approach for discovering unknown reaction pathways from concentration data. The parameters in CRNN correspond to the stoichiometric coefficients and reaction rates and the network has only one hidden layer with the exponential activation functions.

Different from CRNN in [15], we use a single matrix of order $n_r \times n_s$ to represent the stoichiometric coefficients for both the forward and reverse reactions by assuming no catalysis reactions. The negative entries in the matrix denote the stoichiometric coefficients for the reactants and the positive for the products.

On the other hand, the reaction rates often differ in a wide range of magnitudes, which causes a lot of troubles in learning the multiscale chemical reactions. To provide some insights into this difficulty, we design a nonlinear regression problem to fit a polynomial with two terms, see (2.1) in Section 2. The given coefficients of the polynomial differ in several orders of magnitudes and the polynomial degree is to be determined. We find numerically that the conventional optimization algorithm usually gets stuck in the local minima and could not find the true solution. Another observation in the numerical experiment is that the learned polynomial degree of the terms with larger coefficient is close to the true solution. Inspired by this observation, we propose a partial-parameters-freezing (PPF) technique to escape from the local minima. Specifically, we perform a round operation on the learned polynomial degree which are close to integer in the optimization process if the loss function does not decrease. The revised algorithm works well for this problem. Some theoretical analysis is also provided to explain the numerical phenomenon.

We then generalize the PPF technique to learn the multiscale chemical reactions. Notice that the stoichiometric coefficients are integers. In the training process, if the loss function stops to decrease, the stoichiometric coefficients which are close to integers are rounded and then frozen afterwards. With such a treatment, the stoichiometric coefficients are gradually

4

determined, the dimension of the searching space is reduced in the training process, and eventually the global mimina can be obtained. Several numerical experiments including the classical Michaelis–Menten kinetics, the hydrogen oxidation reactions and the simplified GRI-3.0 mechanism verify that our method performs much better in learning the mutiscale chemical reactions.

This paper is organized as follows. In Section 2, we investigate a multiscale nonlinear regression problem numerically and theoretically. Our algorithm for learning the multiscale chemical reactions is presented in Section 3. In Section 4, the performance of the algorithm is validated through several numerical examples. Finally, conclusions and the outlook of future work are presented in Section 5.

2 Multiscale nonlinear regression problem

To provide some insights into the difficulties in learning the multiscale chemical reactions, we consider a nonlinear regression problem to fit the following function:

$$y = f(x; \theta_1, \theta_2) = c_1 x^{\theta_1} + c_2 x^{\theta_2}. \tag{2.1}$$

Here c_1 and c_2 are two given constants satisfying $|c_1| \ll |c_2|$, and θ_1, θ_2 are two integers to be determined. This simple toy model captures two key features of the multiscale chemical reactions. The first feature is that the right-hand side of the chemical reaction ODEs (1.1) is polynomials and the stoichiometric coefficients are integers. The second one is that the multiscale chemical reactions often have reaction rates which differ in several orders of magnitudes.

Given the dataset $\{(x_i, y_i) : i = 1, \cdots, N\}$, we define the loss function to be the mean squared error (MSE):

$$\mathcal{L}(\theta_1, \theta_2) = \frac{1}{N} \sum_{i=1}^{N} (f(x_i; \theta_1, \theta_2) - y_i)^2, \tag{2.2}$$

5

to estimate the parameters θ_1 and θ_2. Next, conventional optimization methods can be used to obtain the estimation of θ_1 and θ_2.

In the numerical experiment, we take $c_1 = 1$ and $c_2 = 100$. The ground truth solutions are $\theta_1 = 1$ and $\theta_2 = 2$. The data x_i for $i = 1, \cdots, N$ are randomly sampled from a uniform distribution in $(0, 1)$ with the number of data $N = 1000$, and $y_i = c_1 x_i + c_2 x_i^2$. The Adam optimization method [17] is applied with the full batch gradient decent. The learning rate is taken to be 10^{-4}. The initial guess of θ_1 and θ_2 is randomly chosen in $(-1, 1)$.

For this toy model, we numerically find that the naive implementation will get stuck in the local minima $(\theta_1, \theta_2) = (3.8286, 1.9745)$ and could not find the true solution. The history of the loss function and the parameters θ_1 and θ_2 in the training is presented in Figure 2.1, see the dashed lines.

Although the naive optimization could not find the global minima, we notice that $\theta_2 = 1.9745$ in this local minima is close to the true solution $\theta_2 = 2$. Inspired by this observation, we propose a partial-parameters-freezing (PPF) technique to escape from the local minima. To be more specific, we keep track of the loss function in the training. If the loss does not decrease, we check the parameters θ_1 and θ_2: if any of these is close to its nearest integer with a given threshold, we round it to the integer and do not update it in the afterwards optimization process.

For comparison, we also plot the history of the loss function and the parameters with the PPF technique in Figure 2.1, see the solid lines. The threshold is taken to be 0.05 in this test. The loss stops decreasing with the epoch around 7000. Then θ_2 is rounded to 2 and only θ_1 is updated afterwards. The true solution is eventually obtained when the epoch is around 10000.

To better understand why it is easy to get stuck in the local minima without the PPF treatment, we investigate the landscape of the loss function. In Figure 2.2, we plot the 3D surface and the contour map for the loss as a function of (θ_1, θ_2). In Figure 2.2 (a), it is
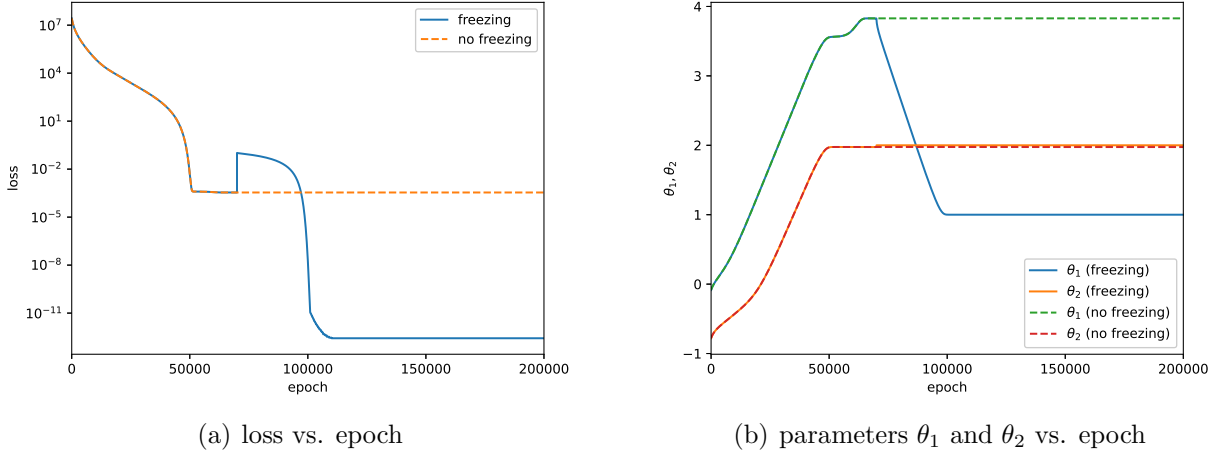
6

(a) loss vs. epoch

(b) parameters θ_1 and θ_2 vs. epoch

Figure 2.1: Multiscale nonlinear regression problem: the history of loss function in (2.2) and the parameters θ_1 and θ_2 in the training process. Solid lines: the method with the PPF technique; dashed lines: the method without the PPF technique.

observed that the loss function has several local minima in which θ_2 is close to 2. Moreover, the local minima $(\theta_1, \theta_2) = (3.8286, 1.9745)$ in the naive implementation is also labeled in Figure 2.2 (b).

We also plot the profiles of the loss function with fixed $\theta_2 = 1.99$, 2 and 2.01 in Figure 2.3. It is observed that slight perturbations in θ_2 have a considerable impact on the minima of the loss function. Moreover, the loss as a 1D function with fixed $\theta_2 = 2$ is well-behaved. This explains why our algorithm is easy to find the global minima after freezing the integer parameter θ_2.

We mention that we also test other cases with different coefficients c_1 and c_2 satisfying $|c_2/c_1| = 10^3, 10^4, 10^5$ and different integers θ_1 and θ_2. The results are similar and thus omitted here.

We conclude this section with some theoretical analysis to explain the local minima phenomenon observed above. By taking gradient of the loss function in (2.2), we have

$$\frac{\partial \mathcal{L}}{\partial \theta_j} = \frac{2c_j c_2}{N} \sum_{i=1}^{N} \left(\frac{c_1}{c_2} (x_i^{\theta_1} - x_i^{\theta_1^e}) + (x_i^{\theta_2} - x_i^{\theta_2^e}) \right) x_i^{\theta_j} \ln x_i, \quad j = 1, 2. \tag{2.3}$$

7

(a) loss function surface plot
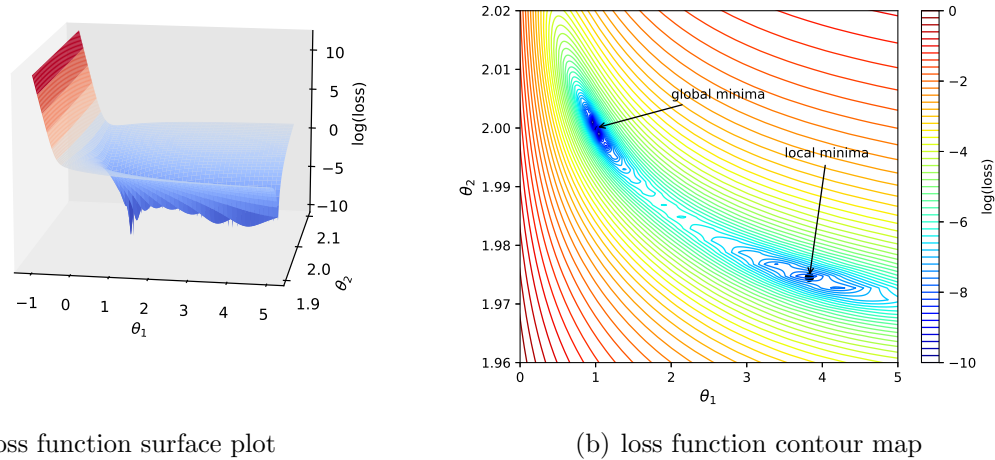
(b) loss function contour map

Figure 2.2: Multiscale nonlinear regression problem: the landscape of the loss function in (2.2). Left: loss function surface plot (in log scale); right: loss function contour map (in log scale), local minima $(\theta_1, \theta_2) = (3.8286, 1.9745)$.
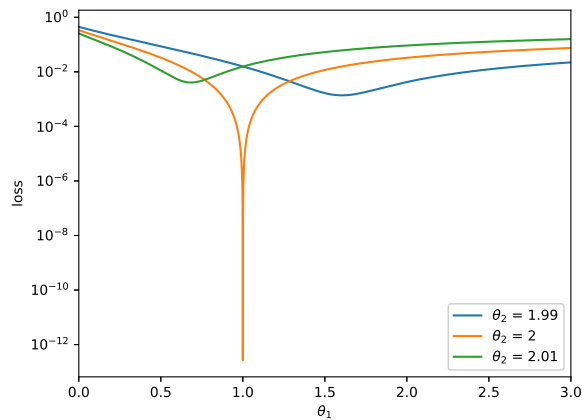

Figure 2.3: Multiscale nonlinear regression problem: loss function in (2.2) with fixed parameters $\theta_2 = 1.99$, 2 and 2.01.

Here θ_i^{e} denotes the true solution of the parameter θ_i for $i = 1, 2$. From the expression (2.3), we can provide some insights on the phenomenon that the local minina θ_2 is close to the true solution θ_2^{e}. To reach the local minima, the gradient should be zero. Refer to the expression (2.3). Since $|c_1/c_2| \ll 1$, whether or not the gradient is close to zero depends mainly on the fact that θ_2, instead of θ_1, is close to the ground truth.

3   Algorithm

In this section, we present our algorithm for learning the multiscale chemical reactions. First, we use a single matrix to represent the stoichiometric coefficients for both the reactants and products. Each row of the matrix represents one reaction, where the negative entries denote the stoichiometric coefficients for the reactants and the positive ones for the products. This setup is valid for systems without catalysis reactions. In addition, we adapt the PPF technique for the multiscale nonlinear regression problem proposed in Section 2 to learn the multiscale chemical reactions (1.1).

We assume that the data are given in the form of the concentrations and the time derivatives in different time snapshots $\{(u_k(t_n), u_k'(t_n)), \ k = 1, \cdots, n_s, \ n = 1, \cdots, N\}$, our goal is to learn the stoichiometric coefficients and the reaction rates. Realistically, often only $u_k(t_n)$ is available, and the time derivatives $u_k'(t_n)$ could be approximated using numerical differentiations [33, 7].

To better illustrate the algorithm, we firstly introduce some vector notations. We denote the forward and reverse reaction rates in (1.1) by $\boldsymbol{k}_f = (k_{1f}, k_{2f}, ..., k_{n_r f})$ and $\boldsymbol{k}_r = (k_{1r}, k_{2r}, ..., k_{n_r r})$. The stoichiometric coefficients in (1.1) are collected in two matrices:

$$\boldsymbol{V}' = \begin{pmatrix} \nu'_{11} & \nu'_{12} & \cdots & \nu'_{1n_s} \\ \nu'_{21} & \nu'_{22} & \cdots & \nu'_{2n_s} \\ \vdots & \vdots & \ddots & \vdots \\ \nu'_{n_r 1} & \nu'_{n_r 2} & \cdots & \nu'_{n_r n_s} \end{pmatrix}, \qquad \boldsymbol{V}'' = \begin{pmatrix} \nu''_{11} & \nu''_{12} & \cdots & \nu''_{1n_s} \\ \nu''_{21} & \nu''_{22} & \cdots & \nu''_{2n_s} \\ \vdots & \vdots & \ddots & \vdots \\ \nu''_{n_r 1} & \nu''_{n_r 2} & \cdots & \nu''_{n_r n_s} \end{pmatrix}. \qquad (3.1)$$

Assume that there is no catalysis reactions. Therefore, only one of ν'_{ik} and ν''_{ik} can be

non-zero for any (i, k). In this case, the matrix $\boldsymbol{V} = (\nu_{ik}) := \boldsymbol{V}'' - \boldsymbol{V}'$ satisfies

$$\nu_{ik} = \begin{cases} \nu_{ik}'', & \text{if} \quad \nu_{ik} \geq 0, \\ -\nu_{ik}', & \text{if} \quad \nu_{ik} < 0. \end{cases}$$

According to this property, we only need to pin down the matrix $\boldsymbol{V}$. Then $\boldsymbol{V}'$ and $\boldsymbol{V}''$ can be recovered by $\nu_{ik}'' = \max(0, \nu_{ik})$ and $\nu_{ik}' = -\min(0, \nu_{ik})$, respectively.

Next, we define the neural network $\mathcal{N} = \mathcal{N}(u_1, \cdots, u_{n_s}) : \mathbb{R}^{n_s} \to \mathbb{R}^{n_s}$ which has the input $\boldsymbol{u} := (u_1, \cdots, u_{n_s})$ and the parameters $\boldsymbol{l}_f = (l_{1f}, l_{2f}, ..., l_{n_r f})$, $\boldsymbol{l}_r = (l_{1r}, l_{2r}, ..., l_{n_r r})$ and $\boldsymbol{V}$:

$$\mathcal{N}(u_1, \cdots, u_{n_s})_k = \sum_{i=1}^{n_r} \nu_{ik} \left(\exp(l_{if}) \prod_{j=1}^{n_s} u_j^{-\min(0, \nu_{ik})} - \exp(l_{ir}) \prod_{j=1}^{n_s} u_j^{\max(0, \nu_{ik})} \right)$$

for $k = 1, \cdots, n_s$. Here the parameters l_{if} and l_{ir} denote the logarithms of the reaction rates k_{if} and k_{ir} [15]. This change of variables technique has two advantages. The first one is that the positivity of the reaction rates is guaranteed automatically. The second one is that the reaction rates for the multiscale chemical reactions usually differ in several orders of magnitudes. The slight changes of l_{if} and l_{ir} will make k_{if} and k_{ir} change a lot, which could potentially make the neural network to be more robust in the training process.

The loss function is defined as the mean squared error (MSE) between the data for the time derivatives and the output of the neural network:

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{n_s} \left(\mathcal{N}(u_1(t_n), \cdots, u_{n_s}(t_n))_k - u_k'(t_n) \right)^2 + \lambda \mathcal{L}_r. \tag{3.2}$$

Here $\lambda \mathcal{L}_r$ is a regularization term with $\lambda > 0$ the regularization constant and

$$\mathcal{L}_r = \sum_{i=1}^{n_r} \sum_{k=1}^{n_s} |\nu_{ik}| + \sum_{i=1}^{n_r} \sum_{k=1}^{n_s} \nu_{ik}^2 + \sum_{i=1}^{n_r} (|l_{if}| + |l_{ir}|) + \sum_{i=1}^{n_r} (l_{if}^2 + l_{ir}^2). \tag{3.3}$$

Here both L_1 and L_2 regularization terms are included.

This neural network works quite well for non-stiff chemical reactions. However, for stiff reactions, we observe that the optimization usually gets stuck in the local minima in the training process and could not find the true solution. The common techniques such as the

mini-batch and reducing the learning rate do not work in such a situation. To attack this problem, we adapt the PPF technique proposed in the previous section.

The training procedure is split into two parts. The first part is to learn the matrix $\boldsymbol{V}$. To better illustrate the algorithm, we introduce some notation. Denote the vector in the j-th row of $\boldsymbol{V}$ by $\boldsymbol{v}_j$ for $j = 1, \cdots, n_r$. Define the distance to the nearest integer for any vector $\boldsymbol{v} \in \mathbb{R}^{n_s}$ as

$$d_{\text{int}}(\boldsymbol{v}) := \|\boldsymbol{v} - \lfloor \boldsymbol{v} \rceil\|_\infty = \max_{i \in \{1, \cdots, n_s\}} |v_i - \lfloor v_i \rceil|, \tag{3.4}$$

where $\lfloor \rceil$ denotes the function rounding an arbitrary real number to its nearest integer and it is defined to work element-wise on vectors. We keep track of the loss function in the training process. If the loss function stops decreasing, we check if any row of $\boldsymbol{V}$ is close to the nearest integers, i.e., $d_{\text{int}}(\boldsymbol{v}_j) \leq \epsilon$. Here, $\epsilon > 0$ is a hyperparameter and we take $\epsilon = 0.05$ in all the numerical examples in Section 4. If the j-th row of $\boldsymbol{V}$ satisfies the condition $d_{\text{int}}(\boldsymbol{v}_j) \leq \epsilon$, then we round $\boldsymbol{v}_j$ to $\lfloor \boldsymbol{v}_j \rceil$ and do not update it in the afterwards training. In addition, to help the optimization algorithm escape from the local minima, we randomly reinitialize other non-integer entries in $\boldsymbol{V}$ when the loss stops decreasing. After all the entries in $\boldsymbol{V}$ reach integer, we freeze them and then learn the parameters $\boldsymbol{l}_f$ and $\boldsymbol{l}_r$ related to the reaction rates. We remark that the SINDy algorithms [4, 13] can also be applied in learning the reaction rates when the stoichiometric coefficients $\boldsymbol{V}$ are known. The algorithm is summarized in Algorithm 1.

Remark 3.1. *Here we assume that all the reactions are reversible. However, the algorithm can be also applied to irreversible reactions without any modification. The expected result is that the learned reverse reaction rates for the irreversible reactions will be close to zero. This will be demonstrated numerically in Example 4.2 in Section 4.*

Remark 3.2. *The number of reactions can be learned by repeatedly executing the algorithm with different n_r. The ground truth of n_r can be inferred from the best one. This will be shown in the numerical examples in the next section.*

Remark 3.3. *In many chemical reaction systems, the rate constants usually depend on the temperature. For example, the Arrhenius law can describe such a dependence:*

$$k = A \exp\left(-\frac{E_a}{RT}\right),$$
$$(3.5)$$

where k is the reaction rate, A is the pre-exponential factor, E_a is the activation energy and R is the gas constant. In this case, the unknown parameters will include the pre-exponential factor, the activation energy and the stoichiometric coefficients. Our PPF technique can be directly applied without much modification. The performance will be verified numerically in the test in the next section.

Algorithm 1: Algorithm for learning chemical reactions

Input : time series data $\{(u_k(t_n), u'_k(t_n)),\ k = 1, \cdots, n_s,\ n = 1, \cdots, N\}$
Output: stoichiometric coefficient matrix $\boldsymbol{V}$, chemical reaction rates $\boldsymbol{k}_f$ and $\boldsymbol{k}_r$

1 initialize hyperparameters: number of reactions n_r, total number of epoch M,
    learning rate lr, regularization coefficient λ, integer threshold ϵ ;

2 initialize parameters: $\boldsymbol{V}$, $\boldsymbol{l}_f$ and $\boldsymbol{l}_r$;
    `// step 1:  learning `V

3 $L_{\text{rec}} = \text{np.zeros}(M)$;   `// record loss function in each epoch`

4 $S_{\text{int}} = [\ ]$;

5 **for** $i = 1, \cdots, M$ **do**

6   Compute loss $\mathcal{L}$;

7   Compute $\frac{\partial \mathcal{L}}{\partial \theta}$ by backpropagation ;

8   Update parameters (excluding the integer entries in $\boldsymbol{V}$) by Adam method ;
        `// if loss increase, then check if any row of `V` is close to integer`

9   **if** $L_{rec}[i] \geq L_{rec}[i-1]$ **then**

10   **for** $j = 1, \cdots, n_r$ **do**

11   **if** $d_{int}(\boldsymbol{v}_j) \leq \epsilon$ **then**

12 $S_{\text{int}}.\text{append}(j)$;

13 $\boldsymbol{v}_j \leftarrow \lfloor \boldsymbol{v}_j \rceil$;

14   **end**

15   **if** $j \notin S_{int}$ **then**

16 $\boldsymbol{v}_j \leftarrow \text{rand}(-2, 2)$; `// random reinitialize non-integer entries in `$V$

17   **end**

18   **end**

19   **end**
        `// if all the entries in `V` are integers, then stop learning `V

20   **if** $S_{int} = \{1, \cdots, n_r\}$ **then**

21   **break**;

22   **end**

23 **end**
    `// step 2:  learning `k_f` and `k_r

24 **for** $i = 1, \cdots, M$ **do**

25   Compute loss $\mathcal{L}$;

26   Compute $\frac{\partial \mathcal{L}}{\partial \theta}$ by backpropagation ;

27   Update parameters θ (excluding $\boldsymbol{V}$) by Adam method ;

28 **end**

29 **for** $i = 1, \cdots, n_r$ **do**

30 $k_{if} \leftarrow \exp(l_{if})$;

31 $k_{ir} \leftarrow \exp(l_{ir})$;

32 **end**

---

4 Numerical results

Here the performance of our algorithm will be shown with five examples. The first example is an artificial reaction mechanism with two reactions [21]. The second one is the well-known Michaelis-Menten kinetics [16] in biochemistry. The third one is the hydrogen oxidation reactions [11, 8]. The fourth one is the extended Zeldovich mechanism, a typical chemical mechanism describing the oxidation of nitrogen and NOx formation [41]. The last one is the simplified GRI-3.0 mechanism, a chemical mechanism describing the methane oxidation [37].

In each numerical example, we randomly take 100 different initial conditions to generate the data. For each initial condition, we take uniform time snapshots at $t_n = n\Delta t$ with $n = 0, \ldots, 10$ and $\Delta t = 0.1$. The data is generated by solving the governing ODEs numerically using implicit Runge-Kutta method of the Radau IIA family of the fifth order [39] with small enough tolerance. The datasets are randomly split into the training datasets and the validation datasets by a ratio of 4:1. It is worthy to note that here we do not take Δt to be too small so that the datasets could be potentially replaced by the experiment data in the future. The algorithm is implemented with PyTorch [28].

Now we present some details of the training and validation for the following four numerical tests. In the training process, all the parameters in the neural network are first randomly initialized from the uniform distribution in the interval $(-0.5, 0.5)$. Then, we update the parameters by minimizing the loss in (3.2) using the standard Adams algorithm [17]. The learning rate is taken to be 10^{-3} and the regularization coefficient λ in (3.2) is 10^{-8}. Recall the training method following (3.4), we take the integer threshold to be 0.05. Besides, the total epoch number is 10^6 and the mini-batch gradient descent is applied with the batch size 10. For the validation, we use the following relative L^2 error:

$$E = \sqrt{\frac{\sum_{n=1}^{N} \sum_{k=1}^{n_s} |\mathcal{N}(u_1(t_n), \cdots, u_{n_s}(t_n))_k - u_k'(t_n)|^2}{\sum_{n=1}^{N} \sum_{k=1}^{n_s} |u_k'(t_n)|^2}}.$$

Here the $(u_k(t_n), u_k'(t_n))$'s come from the validation dataset. For the other details, we refer the

interested readers to our code in `https://github.com/JuntaoHuang/multiscale-chemical-reaction`.

Example 4.1 (hypothetical stiff reaction network). The first test case is an artificial reaction network with two reactions, taken from [21]:

$$F \underset{k_1^-}{\overset{k_1^+}{\rightleftharpoons}} R \tag{4.1a}$$

$$R \underset{k_2^-}{\overset{k_2^+}{\rightleftharpoons}} P \tag{4.1b}$$

Here F, R and P indicate the fuel, radical and product in combustions, respectively. The reaction rates are taken to be $k_1^+ = k_1^- = 1$ and $k_2^+ = k_2^- = 10^3$. The two reversible reactions in (4.1) have dramatically different reaction rates. Thus, the second reaction (4.1b) will quickly approach to equilibrium after a transient period, after which the first one (4.1a) becomes rate-limiting. This simple model is chosen to test the correctness of our code for stiff reactions.

The corresponding ODE system for (4.1) is linear. The eigenvalues of the coefficient matrix are $\lambda_1 = -2000$, $\lambda_2 = -1.5$ and $\lambda_3 = 0$, which differ in several orders of magnitudes. This indicates that the ODE system is stiff [39].

To illustrate the advantage of the PPF technique, we compare the performance of the algorithm with and without this technique. The history of the training and validation errors is shown in Figure 4.4. The relative error stays around 10^{-3} without this technique, and decreases to 10^{-6} after applying this technique. The learned parameters are listed in Table 4.1. The upper part of the table is the learned parameters with the PPF technique, which agrees well with the ground truth in (4.1). By contrast, the algorithm without imposing this technique could not generate the correct result. Moreover, it is interesting to see that, without using the technique, the learned stoichiometric coefficients in the first reaction and the second one has the opposite sign. We also notice that the summation of the forward rate k_f of the first reaction and the reverse rate k_r of the second one is close to the true reaction rate 10^3. The same holds true for the reverse rate of the first reaction and the forward rate of the second one. This indicates that the effect of these two learned reactions is identical to the
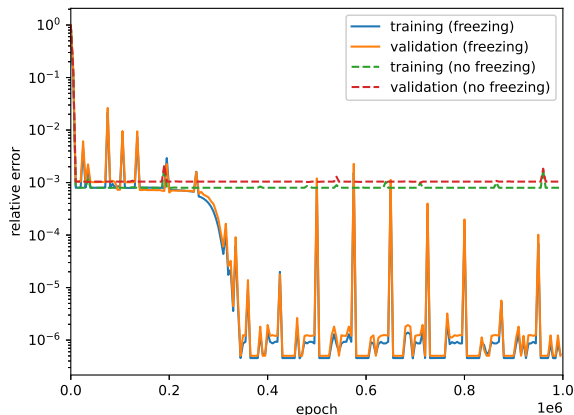
Figure 4.4: Example 4.1: the history of the relative error for the training data and the verification data. Solid line: the method with the PPF technique; dashed line: the method without the PPF technique.

fast reaction (4.1b) and the slow reaction (4.1a) is not captured here. This is similar to the phenomenon we observed in the multiscale nonlinear regression problem in Section 2.

freezing	x_1	x_2	x_3	k_f	k_r
1	0.000	1.000	-1.000	1.000e+03	1.000e+03
2	-1.000	1.000	0.000	1.000e+00	1.000e+00
no freezing	x_1	x_2	x_3	k_f	k_r
1	-0.001	0.999	-0.999	7.448e+02	5.731e+02
2	0.000	-0.999	0.999	4.277e+02	2.559e+02

Table 4.1: Example 4.1: learned parameters. Upper part: with the PPF technique; lower part: without the PPF technique. Here (x_1, x_2, x_3) denotes the row vector of the matrix $\boldsymbol{V}$.

Next, we test the algorithm with different number of chemical reactions. We take the number of reactions ranging from 1 to 4. The relative errors in the training data and the validation data are shown in Figure 4.5. The relative error decreases by three magnitudes when increasing the number of proposed reactions from one to two and reaches a plateau after that. Moreover, it is observed from Table 4.2 that some of the learned stoichiometric coefficients or reaction rates are close to zero if the number of reactions are larger than two. It then can be inferred that the kinetics could be well described with two reactions.

reaction num 1	x_1	x_2	x_3	k_f	k_r
1	0.000	-1.000	1.000	1.000e+03	1.000e+03
reaction num 2	x_1	x_2	x_3	k_f	k_r
1	0.000	1.000	-1.000	1.000e+03	1.000e+03
2	-1.000	1.000	0.000	1.000e+00	1.000e+00
reaction num 3	x_1	x_2	x_3	k_f	k_r
1	0.000	1.000	-1.000	1.000e+03	1.000e+03
2	-1.000	1.000	0.000	9.217e+02	1.218e+02
3	0.750	-0.101	0.384	7.887e$-$04	1.813e$-$03
reaction num 4	x_1	x_2	x_3	k_f	k_r
1	0.000	-1.000	1.000	1.000e+03	1.000e+03
2	0.000	0.000	0.000	5.931e+01	5.929e+01
3	0.000	0.000	0.000	2.526e+01	2.524e+01
4	1.000	-1.000	0.000	1.000e+00	1.000e+00

Table 4.2: Example 4.1: learned parameters with different number of reactions. Here (x_1, x_2, x_3) denotes the row vector of the matrix $\boldsymbol{V}$.
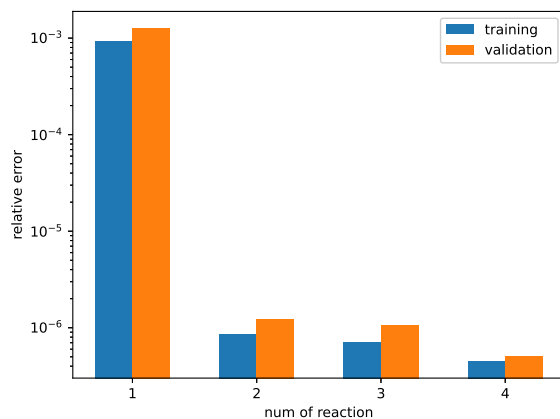

Figure 4.5: Example 4.1: relative error for the training data and the validation data with different number of reactions.

Example 4.2 (enzyme kinetics)**.** In this example, we consider the Michaelis–Menten kinetics [16], one of the best-known models of enzyme kinetics in biochemistry. It involves an enzyme E, binding to a substrate S, to form a complex ES, which in turn releases a product P, regenerating the original enzyme. This can be represented schematically as [16]

$$\text{E} + \text{S} \underset{k_r}{\overset{k_f}{\rightleftharpoons}} \text{ES} \xrightarrow{k_{cat}} \text{E} + \text{P} \tag{4.2}$$
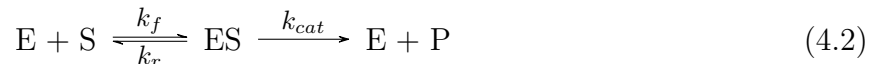
Here k_f denotes the forward rate constant, k_r the reverse rate constant, and k_{cat} the catalytic rate constant. This model is used in a variety of biochemical situations other than enzyme-substrate interaction, including antigen–antibody binding, DNA-DNA hybridization, and protein–protein interaction [26]. Moreover, the reaction rates vary widely between different enzymes. In our test case, we follow [36] and take $k_f = 10^6$, $k_r = 10^3$ and $k_{cat} = 10$.

Note that the second reaction in (4.2) is not reversible. Here, we show that the exactly same algorithm can be applied to this situation. The results with and without the PPF technique are listed in Table 4.3. In the upper part of the table, the reverse rate for the second reaction is 1.949×10^{-4}. It then can be inferred that the system could be well described using two reactions with the second one to be irreversible. Again, the algorithm without this treatment could only get the correct result for the first faster reaction in (4.2). The evolution of the loss function is similar to that in Example 4.1 and thus omitted here.

freezing	x_1	x_2	x_3	x_4	k_f	k_r
1	−1.000	−1.000	1.000	0.000	1.000e+06	1.000e+03
2	1.000	0.000	−1.000	1.000	1.000e+01	1.949e-04
no freezing	x_1	x_2	x_3	x_4	k_f	k_r
1	−0.999	−0.999	0.989	0.000	9.921e+05	9.929e+02
2	−1.001	−1.000	2.385	0.000	7.956e+03	1.392e+01

Table 4.3: Example 4.2: learned parameters. Upper part: with the PPF technique; lower part: without the PPF technique. Here (x_1, x_2, x_3, x_4) denotes the row vector of the matrix $\boldsymbol{V}$.

Next, we test the performance of the algorithm when the reaction rates depend on tem-

perature. We assume that the rate constants in (4.2) satisfy the Arrhenius law:

$$k_f = A_f \exp\left(-\frac{E_{a,f}}{RT}\right), \quad k_r = A_r \exp\left(-\frac{E_{a,r}}{RT}\right), \quad k_{cat} = A_{cat} \exp\left(-\frac{E_{a,cat}}{RT}\right) \qquad (4.3)$$

where the pre-exponential factors are given by

$$A_f = 1, \quad A_r = 4, \quad A_{cat} = 10^3 \qquad (4.4)$$

and the activation energy are

$$E_f = 1600, \quad E_r = 3680, \quad E_{cat} = 2240 \qquad (4.5)$$

and the gas constant $R = 8.3145$. The temperature is randomly taken in a uniform distribution in the interval $[200, 400]$. In this case, the unknown parameters will include the pre-exponential factor, the activation energy in the Arrhenius law, and the stoichiometric coefficients. Our PPF technique can be directly applied without much modification.

We compare the performance of the algorithm with and without the PPF technique. The history of the relative error for the training data and the verification data with variable temperature is shown in Figure 4.6. We see clearly that the errors with the PPF technique are much smaller than those without the technique. We also show the learned parameters in Table 4.4. The upper part of the table is the learned parameters with the PPF technique, which agrees well with the ground truth. By contrast, the algorithm without imposing this technique could not generate the correct result.

Example 4.3 (hydrogen oxidation reaction)**.** In this example, we consider a model for hydrogen oxidation reaction where six species H_2 (hydrogen), O_2 (oxygen), H_2O (water), H, O, OH (radicals) are involved in six steps in a closed system under constant volume and temperature
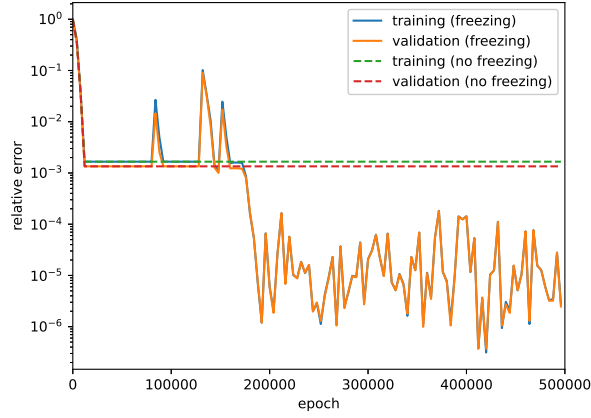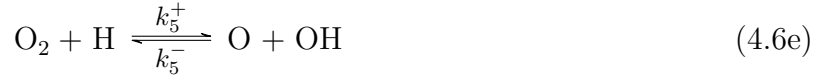
Figure 4.6: Example 4.2: the history of the relative error for the training data and the verification data with variable temperature. Solid line: the method with the PPF technique; dashed line: the method without the PPF technique.

freezing	x_1	x_2	x_3	x_4	A_f	A_r	E_f	E_r
1	1.000	0.000	-1.000	1.000	1.000e+03	1.801e-05	2.240e+03	5.203e+03
2	-1.000	-1.000	1.000	0.000	1.000e+00	4.000e+00	1.600e+03	3.680e+03
no freezing	x_1	x_2	x_3	x_4	A_f	A_r	E_f	E_r
1	1.061	0.001	-1.000	2.887	3.391e+02	1.665e-06	2.240e+03	2.244e+03
2	0.969	0.002	-1.000	0.032	6.640e+02	2.689e-01	6.727e+01	6.684e+02

Table 4.4: Example 4.2: learned parameters with variable temperatures. Upper part: with the PPF technique; lower part: without the PPF technique. Here (x_1, x_2, x_3, x_4) denotes the row vector of the matrix $\boldsymbol{V}$.

[11, 8]:

$$H_2 \underset{k_1^-}{\overset{k_1^+}{\rightleftharpoons}} 2\,H \tag{4.6a}$$

$$O_2 \underset{k_2^-}{\overset{k_2^+}{\rightleftharpoons}} 2\,O \tag{4.6b}$$

$$H_2O \underset{k_3^-}{\overset{k_3^+}{\rightleftharpoons}} H + OH \tag{4.6c}$$

$$H_2 + O \underset{k_4^-}{\overset{k_4^+}{\rightleftharpoons}} H + OH \tag{4.6d}$$

$$O_2 + H \underset{k_5^-}{\overset{k_5^+}{\rightleftharpoons}} O + OH \tag{4.6e}$$

$$H_2 + O \underset{k_6^-}{\overset{k_6^+}{\rightleftharpoons}} H_2O \tag{4.6f}$$

with the reaction rates $k_1^+ = 2$, $k_2^+ = k_3^+ = 1$, $k_4^+ = k_5^+ = 1 \times 10^3$, $k_6^+ = 1 \times 10^2$, $k_1^- = 2.16 \times 10^2$, $k_2^- = 3.375 \times 10^2$, $k_3^- = 1.4 \times 10^3$, $k_4^- = 1.08 \times 10^4$, $k_5^- = 3.375 \times 10^4$, $k_6^- = 7.714285714285716 \times 10^{-1}$. The system (4.6) corresponds to the simplified picture of this chemical process and the reaction rates reflect only orders of magnitude for relevant real-word systems. The magnitude of the reaction rates vary from 10^{-1} to 10^4, which leads to the multiscale phenomena. This reaction network has much more reactions and is more realistic than the first two test cases.

We first compare the performance of our algorithm with and without the PPF treatment. The history of the training and the validation error is shown in Figure 4.7. Again, we observe that this technique greatly reduces the training and validation errors. The learned parameters are listed in Table 4.5. The algorithm can generate the correct result with this technique. On the other hand, without using this technique, the phenomenon of the opposite signs observed in Table 4.1 also appears.

We also test the performance of the algorithm with Gaussian noise. The algorithm can get the correct prediction of the stoichiometric coefficients with the noise level 10^{-4} and 10^{-3}. The learned reaction rates with noise are shown in Table 4.6. The relative errors for reaction rates are typically less than the order of 10^{-2} for 10^{-3} noise and 10^{-3} for 10^{-4} noise.

Moreover, we plot the evolution of the concentrations of the six species with the noise level
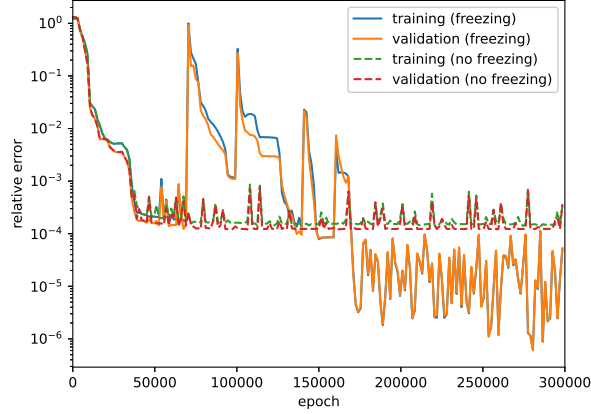
21

Figure 4.7: Example 4.3: the history of the relative error for the training data and the verification data. Solid line: the method with the PPF technique; dashed line: the method without the PPF technique.

freezing	x_1	x_2	x_3	x_4	x_5	x_6	k_f	k_r
1	0.000	1.000	0.000	1.000	-1.000	-1.000	3.375e+04	1.000e+03
2	1.000	0.000	0.000	-1.000	1.000	-1.000	1.080e+04	1.000e+03
3	0.000	0.000	1.000	-1.000	0.000	-1.000	1.400e+03	1.000e+00
4	0.000	1.000	0.000	0.000	-2.000	0.000	3.375e+02	1.000e+00
5	1.000	0.000	0.000	-2.000	0.000	0.000	2.160e+02	2.000e+00
6	-1.000	0.000	1.000	0.000	-1.000	0.000	1.000e+02	7.714e$-$01
no freezing	x_1	x_2	x_3	x_4	x_5	x_6	k_f	k_r
1	0.000	0.938	0.000	0.923	-1.000	-1.001	1.971e+04	5.512e+02
2	0.000	1.087	0.000	1.108	-0.999	-0.999	1.407e+04	4.488e+02
3	0.885	0.000	0.115	-1.000	0.885	-1.000	1.217e+04	2.112e+01
4	-1.004	0.000	0.087	0.910	-1.008	0.913	1.077e+03	2.926e+01
5	0.000	0.996	0.008	0.000	-1.997	0.000	3.379e+02	8.912e$-$01
6	0.987	0.000	0.007	-1.984	0.004	0.004	2.170e+02	3.377e+00

Table 4.5: Example 4.3: learned parameters. Upper part: with the PPF technique; lower part: without the PPF technique. Here $(x_1, x_2, x_3, x_4, x_5, x_6)$ denotes the row vector of the matrix $\boldsymbol{V}$.

22

noise 10^{-3}	k_f	relative error	k_r	relative error
1	3.375e+04	5.706e−05	1.002e+03	2.097e−03
2	1.080e+04	2.789e−04	1.001e+03	1.374e−03
3	1.399e+03	4.413e−04	9.631e−01	3.836e−02
4	3.399e+02	7.103e−03	9.235e−01	8.278e−02
5	2.161e+02	6.927e−04	2.130e+00	6.107e−02
6	9.764e+01	2.417e−02	8.047e−01	4.137e−02
noise 10^{-4}	k_f	relative error	k_r	relative error
1	3.375e+04	6.482e−06	1.000e+03	2.099e−04
2	1.080e+04	2.803e−05	1.000e+03	1.379e−04
3	1.400e+03	4.491e−05	9.963e−01	3.707e−03
4	3.377e+02	7.161e−04	9.923e−01	7.717e−03
5	2.160e+02	6.965e−05	2.013e+00	6.469e−03
6	9.976e+01	2.366e−03	7.748e−01	4.294e−03

Table 4.6: Example 4.3: learned reaction rates with noise.

10^{-3} in Figure 4.8. We observe a good agreement of the solution generated by our learned model and the exact solution. We also measure the prediction errors of the learned model at 100 uniformly points in the time interval $[0, 10]$. The prediction errors are 1.953×10^{-6} with zero noise, 9.152×10^{-4} with noise level 10^{-4} and 8.710×10^{-4} with noise level 10^{-3}.

Example 4.4 (extended Zeldovich mechanism). In this example, we test our algorithm on the extended Zeldovich mechanism, which is a chemical mechanism describing the oxidation of nitrogen and NOx formation [41]. Similar to Example 4.3, this is another realistic test case. The reaction mechanisms read as

$$\text{N}_2 + \text{O} \xrightleftharpoons[k_1^-]{k_1^+} \text{NO} + \text{N} \tag{4.7a}$$

$$\text{N} + \text{O}_2 \xrightleftharpoons[k_2^-]{k_2^+} \text{NO} + \text{O} \tag{4.7b}$$

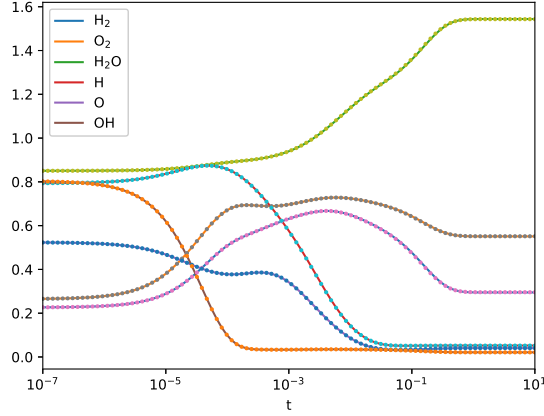$$\text{N} + \text{OH} \xrightleftharpoons[k_3^-]{k_3^+} \text{NO} + \text{H} \tag{4.7c}$$

Figure 4.8: Example 4.3: the evolution of the concentration of the 6 species in the hydrogen oxidation reaction problem obtained by solving the original ODEs (4.6) and our learned ODEs. noise level 10^{-3}.

and the reaction rates are given by the Arrhenius law [12]:

$$k_1^+ = 1.8 \times 10^{11} \exp(-38370/T), \quad k_1^- = 3.8 \times 10^{10} \exp(-425/T),$$

$$k_2^+ = 1.8 \times 10^7 \exp(-4680/T), \quad k_2^- = 3.8 \times 10^6 \exp(-20820/T), \tag{4.8}$$

$$k_3^- = 7.1 \times 10^{10} \exp(-450/T), \quad k_3^- = 1.7 \times 10^{11} \exp(-24560/T),$$

with T the temperature.

In the numerical test, we fix the temperature to be $T = 3000$, which is a reasonable temperature in real applications [12]. At this temperature, the reaction rates are

$$k_1^+ = 5.019 \times 10^5, \quad k_2^+ = 3.782 \times 10^6, \quad k_3^+ = 6.111 \times 10^{10},$$

$$k_1^- = 3.298 \times 10^{10}, \quad k_2^- = 3.679 \times 10^3, \quad k_3^- = 4.732 \times 10^7. \tag{4.9}$$
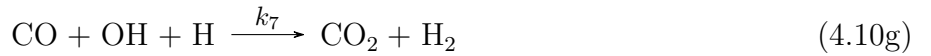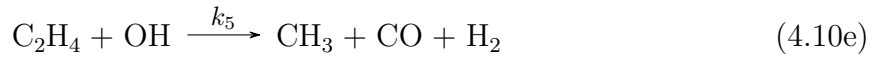
Then, we follow the same procedure in the previous examples to generate the data and execute the algorithm to discover the stoichiometric coefficients and the reaction rates. Again, the algorithm with the PPF treatment can predict the correct result, which is shown in Table 4.7. We observe that the accurate reaction rates are obtained.

Example 4.5 (simplified GRI-3.0 mechanism)**.** In this example, we test our algorithm on the simplified GRI-3.0 mechanism, which is a chemical mechanism describing the methane oxida-

24

freezing	x_1	x_2	x_3	x_4	x_5	x_6	x_7	k_f	k_r
1	0.000	0.000	1.000	−1.000	0.000	−1.000	1.000	6.111e+10	4.732e+07
2	1.000	1.000	−1.000	−1.000	0.000	0.000	0.000	3.298e+10	5.019e+05
3	0.000	1.000	1.000	−1.000	−1.000	0.000	0.000	3.782+06	3.931e+03

Table 4.7: Example 4.4: learned parameters with the PPF technique. Here $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ denotes the row vector of the matrix $\boldsymbol{V}$.

tion [37]. This is the most complicated reaction system tested in the paper. The mechanism includes 16 species with 12 reactions and reads as

$$CH_4 + H \xrightarrow{k_1} CH_3 + H_2 \tag{4.10a}$$

$$CH_2O + H_2 \xrightarrow{k_2} CH_3 + OH \tag{4.10b}$$

$$CH_2O \xrightarrow{k_3} CO + H_2 \tag{4.10c}$$

$$C_2H_6 \xrightarrow{k_4} C_2H_4 + H_2 \tag{4.10d}$$

$$C_2H_4 + OH \xrightarrow{k_5} CH_3 + CO + H_2 \tag{4.10e}$$

$$2\,CO + H_2 \xrightarrow{k_6} C_2H_2 + O_2 \tag{4.10f}$$

$$CO + OH + H \xrightarrow{k_7} CO_2 + H_2 \tag{4.10g}$$

$$H + OH \xrightarrow{k_8} H_2O \tag{4.10h}$$

$$2\,H + 2\,OH \xrightarrow{k_9} 2\,H_2 + O_2 \tag{4.10i}$$

$$H_2 \xrightarrow{k_{10}} 2\,H \tag{4.10j}$$

$$H_2 + O_2 \xrightarrow{k_{11}} HO_2 + H \tag{4.10k}$$

$$H_2O_2 + H \xrightarrow{k_{12}} H_2 + HO_2 \tag{4.10l}$$

The reaction rates are given in [37], which are derived from the reaction rates of the standard GRI-3.0 Mech [35]. We compute the reaction rates with the temperature $T = 3000$ and list them in Table 4.8. Here, the reaction rates are normalized such that the smallest one is of order 1.

Note that all the reactions in (4.10) are not reversible. Here, we apply exactly the same

k_1	k_2	k_3	k_4	k_5	k_6
5.088e+00	1.891e+00	2.607e+00	6.268e+00	5.446e+00	1.283e+01
k_7	k_8	k_9	k_{10}	k_{11}	k_{12}
1.349e+00	5.264e+03	3.268e+01	4.873e+03	2.978e+02	5.227e+03

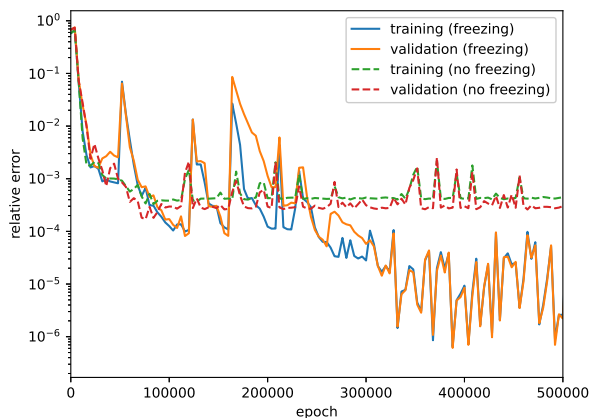Table 4.8: Example 4.5: reaction rates in simplified GRI-3.0 Mech

Figure 4.9: Example 4.5: the history of the relative error for the training data and the verification data. Solid line: the method with the PPF technique; dashed line: the method without the PPF technique.

algorithm to this situation, similar to Example 4.2. To illustrate the advantage of the PPF technique, we first compare the performance of the algorithm with and without this technique. The history of the training and validation errors is shown in Figure 4.9. The relative error stays around 10^{-3} without this technique, and decreases to 10^{-6} after applying this technique.

We also list the learned parameters with the PPF technique in Table 4.9. Here, the learned stoichiometric coefficients are the same with the true coefficients in (4.10) and they are omitted here. The upper part of the table is the learned rates in the forward reactions with the PPF technique, which agrees well with the ground truth in Table 4.8. The learned rates in the reverse reactions are in the magnitude of 10^{-7} to 10^{-4}. It then can be inferred that the system can be well described using only forward reactions. By contrast, the algorithm

k_1^+	k_2^+	k_3^+	k_4^+	k_5^+	k_6^+
5.088e+00	1.891e+00	2.607e+00	6.268e+00	5.446e+00	1.283e+01
k_7^+	k_8^+	k_9^+	k_{10}^+	k_{11}^+	k_{12}^+
1.349e+00	5.264e+03	3.268e+01	4.873e+03	2.978e+02	5.227e+03
k_1^-	k_2^-	k_3^-	k_4^-	k_5^-	k_6^-
2.546e-04	1.695e-04	1.091e-04	1.751e-04	1.103e-04	9.052e-06
k_7^-	k_8^-	k_9^-	k_{10}^-	k_{11}^-	k_{12}^-
8.462e-05	1.146e-05	5.472e-04	2.625e-07	8.566e-07	3.863e-04

Table 4.9: Example 4.5: learned reaction rates in simplified GRI-3.0 Mech. Upper part: reaction rates in the forward reaction; lower part: reaction rates in the reverse reaction.

without imposing this technique could not generate the correct result and we omit the results here.

5 Conclusion

In this paper, we propose a data-driven method to discover multiscale chemical reactions governed by the law of mass action. The method mainly contains two novel points. First, we use a single matrix to represent the stoichiometric coefficients for both the reactants and products in a system without catalysis reactions. The negative entries in the matrix denote the stoichiometric coefficients for the reactants and the positive ones for the products. Second, by considering a multiscale nonlinear regression problem, we find that the conventional optimization methods usually get stuck in the local minima and could not find the true solution. To escape from the local minima, we propose a PPF technique. Notice that the stoichiometric coefficients are integers. In the training process, if the loss function stops to decrease, the stoichiometric coefficients which are close to integers are rounded and then frozen afterwards. With such a treatment, the stoichiometric coefficients are gradually determined, the dimension of the searching space is reduced in the training process, and eventually the global mimina can be obtained. Several numerical experiments including the classical Michaelis–Menten kinetics, the hydrogen oxidation reactions and simplified GRI-3.0 mechanism verify the validity of our

algorithm in learning the multiscale chemical reactions.

There are still some problems to be addressed in order to develop a robust and general framework for discovering multiscale chemical reactions from data. We shall highlight some of the challenges that could guide future advances. First, it is interesting to generalize the PPF technique to the catalysis reactions. Second, the number of species n_s cannot be determined from our algorithm. In principle, to infer the unknown chemical reaction systems, we should have the concentration time series data for all the species. Our algorithm cannot treat the problem when the concentrations for some partial species are unknown. This difficulty may be overcomed by combining the current algorithm with the Neural ODE approach in [15]. The third challenge is that for very complex reaction networks with large number of reactions (hundreds or thousands), our algorithm may not always find out the correct solution. New ideas are needed at this point.

References

[1] B. Bhadriraju, M. S. F. Bangi, A. Narasingam, and S. I. Kwon. Operable adaptive sparse identification of systems (OASIS): application to chemical processes. *AIChE Journal*, 66(11):e16980, 2020.

[2] B. Bhadriraju, A. Narasingam, and J. S.-I. Kwon. Machine learning-based adaptive model identification of systems: Application to a chemical process. *Chemical Engineering Research and Design*, 152:372–383, 2019.

[3] S. L. Brunton, B. R. Noack, and P. Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.

[4] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National*

Academy of Sciences, 113(15):3932–3937, 2016.

[5] S. C. Burnham, D. P. Searson, M. J. Willis, and A. R. Wright. Inference of chemical reaction networks. *Chemical Engineering Science*, 63(4):862–873, 2008.

[6] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

[7] R. Chartrand. Numerical differentiation of noisy, nonsmooth data. *ISRN Applied Mathematics*, 2011, 2011.

[8] E. Chiavazzo and I. V. Karlin. Quasi-equilibrium grid algorithm: Geometric construction for model reduction. *Journal of Computational Physics*, 227(11):5535–5560, 2008.

[9] B. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. N. Kutz, and S. Brunton. PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software*, 5(49):1–4, 2020.

[10] C. W. Gao, J. W. Allen, W. H. Green, and R. H. West. Reaction mechanism generator: Automatic construction of chemical kinetic mechanisms. *Computer Physics Communications*, 203:212–225, 2016.

[11] A. N. Gorban and I. V. Karlin. *Invariant manifolds for physical and chemical kinetics*, volume 660. Springer Science & Business Media, 2005.

[12] R. K. Hanson and S. Salimian. Survey of rate constants in the n/h/o system. In *Combustion chemistry*, pages 361–421. Springer, 1984.

[13] M. Hoffmann, C. Fröhner, and F. Noé. Reactive SINDy: Discovering governing reactions from concentration data. *The Journal of Chemical Physics*, 150(2):025101, 2019.

[14] J. Huang, Z. Ma, Y. Zhou, and W.-A. Yong. Learning thermodynamically stable and galilean invariant partial differential equations for non-equilibrium flows. *Journal of Non-Equilibrium Thermodynamics*, 2021.

[15] W. Ji and S. Deng. Autonomous discovery of unknown reaction pathways from data by chemical reaction neural network. *The Journal of Physical Chemistry A*, 125(4):1082–1092, 2021.

[16] J. P. Keener and J. Sneyd. *Mathematical Physiology*, volume 1. Springer, 1998.

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] D. Langary and Z. Nikoloski. Inference of chemical reaction networks based on concentration profiles using an optimization framework. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(11):113121, 2019.

[19] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[20] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis. DeepXDE: A deep learning library for solving differential equations. *arXiv preprint arXiv:1907.04502*, 2019.

[21] T. Lu and C. K. Law. On the applicability of directed relation graphs to the reduction of reaction mechanisms. *Combustion and Flame*, 146(3):472–483, 2006.

[22] B. Lusch, J. N. Kutz, and S. L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):1–10, 2018.

[23] S. Maddu, B. L. Cheeseman, C. L. Müller, and I. F. Sbalzarini. Learning physically consistent mathematical models from data using group sparsity. *arXiv preprint arXiv:2012.06391*, 2020.

[24] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Inferring biological networks by sparse identification of nonlinear dynamics. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 2(1):52–63, 2016.

[25] T. Nagy, J. Tóth, and T. Ladics. Automatic kinetic model generation and selection based on concentration versus time curves. *International Journal of Chemical Kinetics*, 52(2):109–123, 2020.

[26] D. L. Nelson, A. L. Lehninger, and M. M. Cox. *Lehninger principles of biochemistry.* Macmillan, 2008.

[27] H. G. Othmer. Analysis of complex reaction networks. *Lecture Notes, School of Mathematics, University of Minnesota*, 2003.

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8026–8037, 2019.

[29] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[30] M. Raissi, A. Yazdani, and G. E. Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

[31] R. Ranade, S. Alqahtani, A. Farooq, and T. Echekki. An ANN based hybrid chemistry framework for complex fuels. *Fuel*, 241:625–636, 2019.

[32] R. Ranade, S. Alqahtani, A. Farooq, and T. Echekki. An extended hybrid chemistry framework for complex hydrocarbon fuels. *Fuel*, 251:276–284, 2019.

[33] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.

[34] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017.

[35] G. P. Smith, D. M. Golden, M. Frenklach, N. W. Moriarty, B. Eiteneer, M. Goldenberg, C. T. Bowman, R. K. Hanson, S. Song, W. C. J. Gardiner, V. V. Lissianski, and Z. Qin. Gri-mech home page, http://www.me.berkeley.edu/gri_mech/.

[36] V. Srinivasan and R. Aiken. Stage-wise parameter estimation for stiff differential equations. *AIChE journal*, 32(2):195–199, 1986.

[37] C. Sung, C. Law, and J.-Y. Chen. Augmented reduced mechanisms for no emission in methane oxidation. *Combustion and Flame*, 125(1-2):906–919, 2001.

[38] E. O. Voit, H. A. Martens, and S. W. Omholt. 150 years of the mass action law. *PLoS Comput Biol*, 11(1):e1004012, 2015.

[39] G. Wanner and E. Hairer. *Solving Ordinary Differential Equations II*. Springer Berlin Heidelberg, 1996.

[40] M. J. Willis and M. von Stosch. Inference of chemical reaction networks using mixed integer linear programming. *Computers & Chemical Engineering*, 90:31–43, 2016.

[41] I. Zeldovich, G. I. Barenblatt, V. Librovich, and G. Makhviladze. Mathematical theory of combustion and explosions. 1985.