A Data-Driven Framework for Computationally Efficient Integration of Chemical Kinetics Using Neural Ordinary Differential Equations

Shubhangi Bansude[1]
Department of Mechanical Engineering,
University of Connecticut,
Storrs, CT 06269
e-mail: shubhangi.bansude@uconn.edu

Farhad Imani
Assistant Professor
Department of Mechanical Engineering,
University of Connecticut,
Storrs, CT 06269
e-mail: farhad.imani@uconn.edu

Reza Sheikhi
Fellow ASME
Department of Mechanical Engineering,
University of Connecticut,
Storrs, CT 06269
e-mail: reza.sheikhi@uconn.edu

A data-driven methodology is introduced for computationally efficient integration of systems of stiff rate equations in chemical kinetics using neural ordinary differential equations (NODE). A systematic algorithm is developed for training data generation and sampling. Subsequently, a novel transformation technique for sampled training data is designed to regularize the neural network parameters, leading to a stable training process. Finally, the NODE network is iteratively trained to learn the accurate neural network representation of chemical kinetics source terms by minimizing the mean absolute error between the true and predicted solutions. The computational efficiency and accuracy of the NODE network are evaluated by simulating the evolution of the thermochemical state of a constant pressure homogeneous hydrogen-air reactor. The combustion of hydrogen in air is described by a finite-rate mechanism including 9 chemical species and 21 reaction steps. The NODE network shows excellent multi-step prediction accuracy for a wide range of initial temperatures and equivalence ratios, spanning the composition space of real flames. The NODE also exhibit a significant reduction in numerical stiffness of the system, enabling the utilization of explicit solvers for integration. The present simulation results using NODE demonstrate up to 70% speed up in computation time compared to direct integration of the chemical mechanism with at most 3.16% relative error in ignition delay time.
[DOI: 10.1115/1.4062105]

Keywords: combustion, computational methods, hydrogen, chemical kinetics, machine learning, neural networks, neural ordinary differential equations

1 Introduction

Reliable simulation of turbulent reacting flows in combustion systems is integral to the efficient design of engineering devices such as internal combustion engines, gas turbines, industrial furnaces, and propulsion systems. These simulations require solving fully coupled transport equations for chemical species, which is often computationally prohibitive due to the numerical stiffness introduced by chemical kinetics. Operator splitting is an approach that can facilitate the solution procedure by separating the mixing process (due to convection and diffusion) from the chemical reaction effects and solving the corresponding equations in two separate steps. The latter presents a system of stiff ordinary differential equations (ODEs) [1–3], whose integration is a primary bottleneck in these simulations because of the high nonlinearity and stiffness resulting from the chemical reaction source term. This issue is further exacerbated by a large number of chemical species (typically of order $10-10^4$) involved in practical combustion applications, which increases the dimensionality of the species composition space [4]. Owing to these challenges, direct integration of chemical kinetics remains infeasible for many practical applications. This can be alleviated using efficient data-driven methods which offer a new way to realize the realistic implementation of combustion chemistry in practice.

On the traditional modeling front, mechanism reduction is one approach to mitigating the computational cost associated with chemistry [5]. Mechanism reduction involves abating the cost of chemistry integration by reducing the size and stiffness of the detailed kinetics mechanism (DKM). Here, the size of the chemical mechanism is decreased through a combination of skeletal and time-scale based reduction methods [5]. Skeletal reduction entails elimination of unimportant species and reactions from DKM through methods such as sensitivity analysis, directed relation graph [6], principal component analysis [7], and computational singular perturbation [8]. Time-scale based reduction consists of reduction in independent variables through quasi-steady-state approximations (QSSA) [9], partial-equilibrium approximations [10], rate-controlled constrained-equilibrium [11–16], and intrinsic low dimensional manifold [17]. Although these approaches aid in mitigating the computational demand, the resulting reduced mechanisms often retain some degree of stiffness, requiring the use of stiff ODE solvers to obtain the solution. Subsequently, in some research studies, skeletal/reduced mechanisms are accompanied by tabulation methods to further alleviate the computational cost in large-scale simulations. Look-up table [18] is one common tabulation strategy where integrals of reaction rates corresponding to the possible composition space are computed in advance and stored in a table. During the simulation, species rates are accessed through straightforward interpolation on table records. While this approach can reduce CPU time, it is only suitable for cases with a small number of species as the memory requirement to store the records grows exponentially with the dimensionality of the composition space. The pivotal approach towards circumventing this problem is in-situ adaptive tabulation (ISAT), proposed by Pope [4]. ISAT employs on-the-fly tabulation of composition space visited during the simulation and retrieval of the states via a binary search tree.

[1]Corresponding author.
Manuscript received February 17, 2023; final manuscript received March 3, 2023; published online April 6, 2023. Assoc. Editor: Hameed Metghalchi.

Despite the reduction of CPU time and memory requirement asymptotically, ISAT necessitates performing direct integration of DKM at the initial stages of the simulation, which can make the tabulation prohibitively demanding for practical applications. Furthermore, as more regions of the composition space are accessed, the storage requirement and average retrieval times to traverse the search tree can increase considerably.

In recent years, data-driven approaches to scientific computing have gained popularity as a method of overcoming computational bottlenecks in traditional modeling approaches. Amongst them, artificial neural networks (ANNs) are particularly lucrative because of their capability to approximate highly nonlinear functions. ANNs are apt for a wide range of tasks, including nonlinear regression, clustering, pattern recognition, and time series prediction. Numerous researchers have devoted their attention to the ANN-based regression approaches for chemistry integration to tackle the high computational cost of direct integration of DKM and elevated storage requirement of tabulation methods [19–28]. Christo et al. [24] used ANN to incorporate chemistry in turbulent simulations using a reduced three-step $H_2/CO_2/O_2$ mechanism. The methodology was further refined by Blasco et al. [19,20] to model more complex methane chemistry with 13 species. A slightly different approach was taken by Chen et al. [21], which involves fitting the results obtained from ISAT with ANN to reduce the storage requirement associated with ISAT. These studies reported good agreements with direct integration calculations in addition to considerable savings in computational time and memory requirements. They however identified a major difficulty in generating the representative datasets, challenging the generalization of ANNs to actual conditions encountered in real flames. Thereafter, Chatzopoulos and Rigopoulos [26] proposed a methodology to generate training data with an abstract problem (such as laminar flamelets) spanning the expected composition space and applying the trained ANNs to simulate real turbulent flames. This concept was further applied to the simulation of more complex Sydney flame L for methane combustion that features significant levels of local extinction and re-ignition [25]. Seemingly, all prior research work predominantly focused on employing feed-forward neural networks as single-step chemistry integrators with only slight differences in their network architectures. The model inputs comprise of the species concentration and enthalpy at a given time, and outputs are the species concentration/chemical source terms at the next time-step. Such ANN models trained via minimizing single-step prediction errors have limited applicability in multi-step predictions. By recursively feeding the outputs from the last time-step to the ANN as new inputs, the model could accumulate larger errors in longer-time predictions. This challenge was addressed in Ref. [29] where the neural network architecture of the residual network model (ResNet) is employed to minimize multi-step prediction error.

The primary objective of the present study is to address the aforementioned challenges by introducing a novel data-driven framework that can further reduce the computational cost associated with the integration of chemical kinetics. This framework is based on a recent class of machine learning algorithms called neural ordinary differential equations (NODE) to build a multi-step chemistry integrator. The NODE algorithm, which is a discrete counterpart of ResNet [30], is a hybrid of neural net and differential equation solvers with several major properties: First, embedding a differential equation solver within the network immensely simplifies its architecture, consequently reducing its memory requirements. Second, the NODE network directly minimizes the difference between the predicted and true solutions at all intermediate points along the solution trajectory, providing better accuracy than ResNet with a similar architecture [30]. Finally, the NODE network can be designed to reduce the stiffness of the original ODE by incorporating non-stiff ODE solvers during the training process. These properties make the NODE network particularly suitable for chemical kinetics integration. The work of Owoyele and Pal [31] is among the first studies to demonstrate the use of the NODE network for combustion chemistry. This study indicated that attempting to learn the source terms for all species concurrently was too unstable because of the nonlinearity associated with the dynamics of various species. Hence, separate networks were trained for each major species and temperature, while minor species were neglected. Considering the strongly coupled nature of chemical kinetics, this may lead to erroneous results when this model is used in computational fluid dynamics calculations. The novelty of the present study lies in devising a systematic approach to pre-process and regularize the data to stabilize the training of NODE and to learn the source terms of all the species with a single network. Besides NODE, Ji et al. [32] demonstrated a different approach of physics-informed neural network (PINN) for chemical kinetics integration [32]. This study elucidated the limitations of using PINN for stiff chemistry. To address the issue of stiffness, at first, the study employed QSSA to relieve the stiffness of the original ODE systems and then applied PINN to the converted non/mild-stiff systems. In the present work, we illustrate an approach to reducing the stiffness of the underlying chemical kinetics ODE by embedding an explicit ODE solver in the NODE framework.

The paper is organized as follows: Sec. 2 outlines the basics of NODE and chemically reactive systems, followed by research methodology to develop an NODE-based chemistry integrator framework. Section 3 discusses the performance and numerical efficiency of the NODE network for hydrogen-air auto-ignition in a canonical zero-dimensional constant pressure homogeneous reactor. Finally, Sec. 4 provides the concluding remarks along with an overview of the capabilities of the proposed methodology.

2 Research Methodology

In the present study, we develop a novel machine learning-assisted chemistry integrator for chemically reactive systems by employing the NODE framework. The primary aim is to speed up the integration of chemical kinetics particularly for turbulent combustion simulations. This is achieved by developing a systematic algorithm for data generation and pre-processing, and designing a stable NODE network to reduce stiffness and control multi-step prediction errors. To take into account species composition space representative of typical flames, a constant pressure homogeneous reactor containing a gas mixture spanning a range of initial conditions is adopted for training the model. The basics of NODE formulation, chemical kinetics specifications of DKM, and NODE network training are explained in the following subsections.

2.1 Basics of NODE. In classical data modeling, for a given set of N pairs of data points $D = \{(x_1, y_1), (x_2, y_2), , (x_N, y_N)\}$, the objective is to find a function that best describes the data. Neural networks consist of a large number of parameters, which make them capable of learning complex patterns found in data accurately. The modeling for the dataset D with machine learning can be broadly categorized into two approaches:

(1) Regression: In this approach, the neural network approximates the direct functional relationship between input domain x and output domain y. This can be achieved by designing a neural network and optimizing its parameters to minimize the loss function L. This function provides a measure of the difference between the true and predicted y values and can be defined as the mean absolute error, mean squared error (MSE), cross-entropy loss, etc. depending on the problem under consideration.

(2) NODE: In this approach, the neural network approximates the rate of change of y with x (i.e., $\frac{dy}{dx}$) as opposed to the direct relationship between x and y. Similar to the regression approach, the objective in NODE is to optimize the network parameters to minimize L. The intermediate (x, y) values are then evaluated by numerically integrating the neural network for a given initial condition using an ODE solver.

Modeling the rate of change through NODE, instead of a direct relationship, reduces the number of parameters of the function describing the data, as well as the number of evaluations required to find the optimal parameters. Furthermore, for faster computations, NODE can be integrated with advanced ODE solvers designed to optimize the number of function evaluations and control the intermediate sub-time-steps to determine the solution with the desired accuracy.

Consider a process governed by a system of ODEs

$$\frac{\mathrm{d}\mathbf{z}(t)}{\mathrm{d}t} = \mathbf{f}(\mathbf{z}(t), t) \tag{1}$$

together with some observations along its trajectory

$$\mathbf{Z} = \{(t_0, \mathbf{z}_0), (t_1, \mathbf{z}_1), (t_2, \mathbf{z}_2), \ldots, (t_N, \mathbf{z}_N)\} \tag{2}$$

In Eq. (1), $\mathbf{z}(t)$ and $\mathbf{f}(\mathbf{z}(t), t)$ are the dependent variable and the exact source term arrays, respectively. The objective of NODE network is to model $\mathbf{f}(\mathbf{z}(t), t)$ with approximate parametric function $\hat{\mathbf{f}}(\mathbf{z}(t), t, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes the network parameters. Consider, for example, a section of $\mathbf{z}(t)$ trajectory between $t = t_i$ and $t = t_{i+1}$. The approximate $\mathbf{z}(t)$ between these two points is found by numerically integrating $\hat{\mathbf{f}}(\mathbf{z}(t), t, \boldsymbol{\theta})$ with initial condition $\mathbf{z}(t_i)$. The scalar loss function L measures the difference between the approximated $\mathbf{z}(t_{i+1})$ and the true $\mathbf{z}_{i+1}$ values

$$L(\mathbf{z}(t_{i+1})) = L\left(\mathbf{z}(t_i) + \int_{t_i}^{t_{i+1}} \hat{\mathbf{f}}(\mathbf{z}(t), t, \boldsymbol{\theta})\,\mathrm{d}t\right) \tag{3}$$

in which the integral operation is performed using an ODE solver. During the training of NODE, network parameters $\boldsymbol{\theta}$ are determined by minimizing the loss function value. This optimization requires the gradients of loss function with respect to its parameters (t, $\mathbf{z}(t)$ and $\boldsymbol{\theta}$). In contrast to regression neural networks, these gradients are not readily available since the dependency of loss function to its parameters is implicit and dynamic through the integration operation in Eq. (3). The main challenge in computing these gradients is performing reverse-mode differentiation or backpropagation through the ODE solver. One approach is to differentiate through the forward pass operations, which is straightforward but incurs high memory demands and introduces additional numerical errors [31]. A more efficient approach is to utilize the adjoint sensitivity method [31]. In this method, the gradients are computed by solving a second augmented ODE system backward in t (i.e., from t_N to t_0). We define the adjoint $\mathbf{a}(t)$ as

$$\mathbf{a}(t) = -\frac{\partial L}{\partial \mathbf{z}(t)} \tag{4}$$

which determines the dependency of L on the intermediate (hidden) states $\mathbf{z}(t)$ reached during the integration. As detailed in Ref. [31], the adjoint is governed by the following adjoint differential expression:

$$\frac{\mathrm{d}\mathbf{a}(t)}{\mathrm{d}t} = -\mathbf{a}(t)\frac{\partial \hat{\mathbf{f}}(\mathbf{z}(t), \boldsymbol{\theta}, t)}{\partial \mathbf{z}(t)} \tag{5}$$

Equation (5) can be integrated backward using an ODE solver starting from the known initial value of $\partial L/\partial \mathbf{z}(t_N)$ to compute all hidden state gradients $\partial L/\partial \mathbf{z}(t)$ back to $\partial L/\partial \mathbf{z}(t_0)$. Besides the adjoint, optimizing L requires the calculation of loss function gradient with respect to $\boldsymbol{\theta}$. This quantity may be written in terms of $\partial L/\partial \mathbf{z}(t)$ as

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \frac{\partial L}{\partial \mathbf{z}(t)}\frac{\partial \mathbf{z}(t)}{\partial \boldsymbol{\theta}} \tag{6}$$

However, $\mathbf{z}(t)$ states are obtained numerically at the integration substeps and their dependency on $\boldsymbol{\theta}$ is not known in advance. Chen et al. [31] resolve this problem by treating the network parameters $\boldsymbol{\theta}$ to be a part of the system dynamics; i.e., $\boldsymbol{\theta}$ is state-dependent and determined alongside $\mathbf{z}(t)$ and t as a part of the numerical integration. Accordingly, $\mathbf{z}(t)$, $\boldsymbol{\theta}(t)$, and $t(t)$ are obtained concurrently by defining an augmented system of ODEs

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{z}\\ \boldsymbol{\theta}\\ t\end{bmatrix}(t) = \hat{\mathbf{f}}_{aug}(\mathbf{z}(t), \boldsymbol{\theta}, t) = \begin{bmatrix}\hat{\mathbf{f}}(\mathbf{z}(t), \boldsymbol{\theta}, t)\\ \mathbf{0}\\ 1\end{bmatrix} \tag{7}$$

The adjoint state corresponding to this augmented ODE is defined as

$$\begin{bmatrix}\mathbf{a}(t) & \mathbf{a}_{\boldsymbol{\theta}}(t) & \mathbf{a}_t(t)\end{bmatrix} = \frac{\partial L}{\partial \begin{bmatrix}\mathbf{z}(t) & \boldsymbol{\theta}(t) & t(t)\end{bmatrix}} \tag{8}$$

Subsequently, the augmented adjoint differential equations corresponding to the augmented ODE in Eq. (7) can be obtained by generalizing Eq. (5) as

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{a}(t) & \mathbf{a}_{\boldsymbol{\theta}}(t) & \mathbf{a}_t(t)\end{bmatrix} = -\begin{bmatrix}\mathbf{a}(t) & \mathbf{a}_{\boldsymbol{\theta}}(t) & \mathbf{a}_t(t)\end{bmatrix}\frac{\partial \hat{\mathbf{f}}_{aug}(\mathbf{z}(t), \boldsymbol{\theta}, t)}{\partial \begin{bmatrix}\mathbf{z}(t) & \boldsymbol{\theta}(t) & t(t)\end{bmatrix}} \tag{9}$$

where the Jacobian is

$$\frac{\partial \hat{\mathbf{f}}_{aug}(\mathbf{z}(t), \boldsymbol{\theta}, t)}{\partial \begin{bmatrix}\mathbf{z}(t) & \boldsymbol{\theta}(t) & t(t)\end{bmatrix}} = \begin{bmatrix}\frac{\partial \hat{\mathbf{f}}}{\partial \mathbf{z}(t)} & \frac{\partial \hat{\mathbf{f}}}{\partial \boldsymbol{\theta}(t)} & \frac{\partial \hat{\mathbf{f}}}{\partial t(t)}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix} \tag{10}$$

Substituting Eq. (10) into Eq. (9), we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{a}(t) & \mathbf{a}_{\boldsymbol{\theta}}(t) & \mathbf{a}_t(t)\end{bmatrix} = -\begin{bmatrix}\mathbf{a}(t)\frac{\partial \hat{\mathbf{f}}_{aug}}{\partial \mathbf{z}(t)} & \mathbf{a}(t)\frac{\partial \hat{\mathbf{f}}_{aug}}{\partial \boldsymbol{\theta}(t)} & \mathbf{a}(t)\frac{\partial \hat{\mathbf{f}}_{aug}}{\partial t(t)}\end{bmatrix} \tag{11}$$

This augmented ODE system is solved backward in t to obtain all derivatives of L: $\mathbf{a}(t) = \partial L/\partial \mathbf{z}(t)$, $\mathbf{a}_{\boldsymbol{\theta}}(t) = \partial L/\partial \boldsymbol{\theta}(t)$, and $\mathbf{a}_t(t) = \partial L/\partial t(t)$. For example, for the part of the trajectory between t_i and t_{i+1}, assuming known initial conditions at t_{i+1} as follows:

$$\begin{aligned}\mathbf{a}(t_{i+1}) &= \frac{\partial L}{\partial \mathbf{z}(t_{i+1})}\\ \mathbf{a}_{\boldsymbol{\theta}}(t_{i+1}) &= 0\\ \mathbf{a}_{\mathbf{t}}(t_{i+1}) &= \frac{\partial L}{\partial \mathbf{z}(t_{i+1})}\frac{\partial \mathbf{z}(t_{i+1})}{\partial t} = \mathbf{a}(t_{i+1})\hat{f}(\mathbf{z}_{i+1}, t_{i+1}, \boldsymbol{\theta})\end{aligned} \tag{12}$$

the derivatives at $t = t_i$ are obtained as

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{z}(t_i)} &= \mathbf{a}(t_{i+1}) + \int_{t_{i+1}}^{t_i} \mathbf{a}(t)\frac{\partial \hat{\mathbf{f}}(\mathbf{z}(t), \boldsymbol{\theta}, t)}{\partial \mathbf{z}}\,\mathrm{d}t\\ \frac{\partial L}{\partial \boldsymbol{\theta}(t_i)} &= \mathbf{a}_{\boldsymbol{\theta}}(t_{i+1}) + \int_{t_{i+1}}^{t_i} \mathbf{a}(t)\frac{\partial \hat{\mathbf{f}}(\mathbf{z}(t), \boldsymbol{\theta}, t)}{\partial \boldsymbol{\theta}}\,\mathrm{d}t\\ \frac{\partial L}{\partial t(t_i)} &= \mathbf{a}_t(t_{i+1}) + \int_{t_{i+1}}^{t_i} \mathbf{a}(t)\frac{\partial \hat{\mathbf{f}}(\mathbf{z}(t), \boldsymbol{\theta}, t)}{\partial t}\,\mathrm{d}t\end{aligned} \tag{13}$$

where the integrals are solved by running the ODE solver backward in t to find the derivatives at t_i, equivalent to differentiation using backpropagation in conventional neural networks. For the entire $z(t)$ trajectory, following this procedure with the initial conditions at t_N, similar to those in Eq. (12), the derivatives at all data points t_{N-1}, …, t_0 can be obtained. The backpropagation along with the forward pass provides the $\mathbf{z}(t)$ states at all intermediate t values.

The presented reverse-mode differentiation through ODE formulation is implemented in `torchdiffeq` library in PyTorch [30,33]. In the present study, we adopt `torchdiffeq` algorithmic package to train the NODE for chemical kinetics as described in the following section.

2.2 Chemical Kinetics Specifications and Training Data Generation. We consider a constant pressure homogeneous reactor to demonstrate the operation of NODE. For this system,

the conservation equations governing time (t) variation of the thermochemical state of the system can be expressed as

$$\frac{d\mathbf{\Phi}(t)}{dt} = \mathbf{S}(\mathbf{\Phi}(t)) \tag{14}$$

where, $\mathbf{\Phi}(t)$ is the vector of dependent variables including mass fractions (MFs) of N_s number of species $[y_1, y_2, \ldots, y_{N_s}]$ and temperature (T), i.e., $\mathbf{\Phi} = [\mathbf{\Phi}_1, \mathbf{\Phi}_2, \ldots, \mathbf{\Phi}_{N_s}, \mathbf{\Phi}_{N_s+1}] \equiv [y_1, y_2, \ldots, y_{N_s}, T]$. The source terms $\mathbf{S}$ are due to chemical reaction (for $\mathbf{\Phi}_1, \mathbf{\Phi}_2, \ldots, \mathbf{\Phi}_{N_s}$) and heat release (for $\mathbf{\Phi}_{N_s+1}$). The chemical kinetics mechanism in this study consists of nine species: H_2, H, O_2, OH, O, H_2O, H_2O_2, HO_2, and N_2 along with 21 reaction steps [34,35].

During the training step, the source term array $\mathbf{S}$ described by DKM is integrated to generate $\mathbf{\Phi}(t)$ as the training data for NODE. To generate the training data, 25 constant pressure homogeneous hydrogen-air reactors, whose temporal thermochemical states are governed by Eq. (14), are considered with a wide range of initial conditions. Equation (14) is integrated in time to generate representative composition space for these reactors. Here, we consider the initial temperature ranging from 1000 K to 1200 K and equivalence ratios in the range $0.5 - 1.5$ at atmospheric pressure (101325 Pa). Solution for each of the 25 cases is advanced from $t = 0$ to 10^{-2} s with fixed $dt = 10^{-6}$ s, generating 10^4 uniformly spaced points along the temporal composition trajectory for each simulation. As a result, a training dataset of 25×10^4 samples is obtained. In this study, we utilize the ODE solver LSODA via `scipy` toolkit [36] for direct integration of Eq. (14) with DKM description of chemistry. LSODA is a hybrid solver, switching automatically between the non-stiff Adams method and the stiff backward differentiation formula (BDF) solver. Cantera [37] open-source software is used for thermodynamics and chemical kinetics calculations. The use of a complete dataset (25×10^4 data points) for training is redundant as most composition states reach equilibrium after the ignition. Hence, we strategically sample a subset from the complete solution and pre-process it to accelerate the training process. Samples of uneven log-spaced 50 points are chosen out of 10^4 time-steps for each simulation. Half of the sampled points are placed before the ignition delay time and the rest after the ignition. The ignition delay time is defined as the instant when the temperature reaches $T_0 + 400$ K. An example of sampling is shown in Fig. 1 for three cases at initial temperatures of $T_0 = 1000$ K, 1100 K, 1200 K, and stoichiometric equivalence ratio. In this study, 50 samples are chosen as they can capture the dynamics of a single ignition curve well with reasonable computation time for training the NODE network. Depending on the desired accuracy and computational resources available a different number of samples can be chosen for the training of the network.

Another challenge with the training datasets for chemical reaction applications is the wide range of species concentration magnitudes; for example, the major species mass fraction values are of order $10^{-3} - 10^{0}$ while those for minor species are in the order $10^{-3} - 10^{-8}$. Therefore, for the stability of the network, all the composition variables are normalized based on the respective minimum $\mathbf{\Phi}_{\min}$ and maximum $\mathbf{\Phi}_{\max}$ values of each scalar obtained from the complete training dataset

$$\overline{\mathbf{\Phi}} = \frac{\mathbf{\Phi} - \mathbf{\Phi}_{\min}}{\mathbf{\Phi}_{\max} - \mathbf{\Phi}_{\min}} \tag{15}$$

where the normalized values $\overline{\mathbf{\Phi}}$ vary in [0, 1] range.

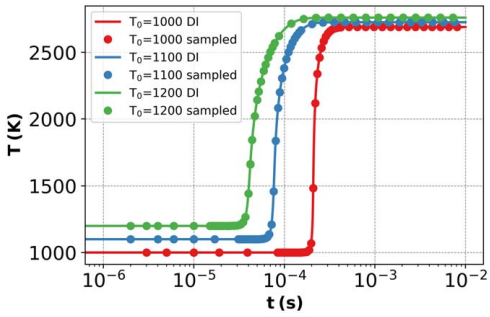

Fig. 1 Depiction of sampled training points at initial temperatures $T_0 = 1000$ K, 1100 K, 1200 K for stoichiometric equivalence ratio in constant pressure homogeneous H_2-air reactor. The lines denote the direct integration of DKM solution for 10^4 time-steps. Symbols denote 50 sampled data points for NODE network training.

2.3 Architecture and Training of NODE Network. The neural ODE network encapsulating Eqs. (14) and (15) is mathematically represented as

$$\frac{d\overline{\mathbf{\Phi}}(t)}{dt} = \mathbf{Net}(\overline{\mathbf{\Phi}}(t), \boldsymbol{\theta}, t) \tag{16}$$

where $\mathbf{Net}(\overline{\mathbf{\Phi}}(t), \boldsymbol{\theta}, t)$ is the representation of the chemical reaction source term of $\overline{\mathbf{\Phi}}(t)$ by the neural network. Subsequent to the training step, integration of this equation followed by denormalization (reverse transformation of Eq. (15)) provides the thermochemical state of the system at time t with the desired accuracy at a reduced computational cost compared to direct integration of DKM according to Eq. (14). Figure 2 shows a graphical representation of NODE chemistry integrator architecture for a single training sample which consists of randomly chosen initial conditions and integration time, i.e., $(t_0, \overline{\mathbf{\Phi}}_0)$ and t_N as input, and corresponding integrated values, i.e., $\overline{\mathbf{\Phi}}_N$ as output. The integrator has two parts: a neural network $\mathbf{Net}(\overline{\mathbf{\Phi}}(t), \boldsymbol{\theta}, t)$ and an embedded ODE solver. The neural network consists of three hidden layers comprising of 150 neurons each. Each hidden layer is followed by a nonlinear exponential linear unit (ELU) activation function. ELU is chosen to mitigate potential issues with vanishing gradients of loss function caused by certain activation functions (e.g., sigmoid function) making the network hard to train. The explicit Runge–Kutta method of order (4)5 (dopri5) [33] with absolute and relative error tolerances 10^{-9} and 10^{-7} is chosen, respectively, for integration during the training. The embedding of an explicit solver ensures that the network learns the non-stiff representation of the chemical reaction source term through training. These error tolerances are selected to match the accuracy level of the ODE solver used during the training data generation. Nevertheless, different tolerance values can be chosen depending on the accuracy needed when utilizing the network after the training.

The network is trained by sending data in batches for better learning performance. A training batch consists of $N = 50$ filtered time instances on the composition space evolution trajectory for $I = 25$ initial conditions. Thus, in every epoch, the network learns from all training points at once. Each epoch consists of a forward pass to calculate the MAE of a training batch, subsequently followed by backpropagation to obtain derivatives of the loss function and to update the network parameters. A NODE network training algorithm is outlined in a pseudo-code in Algorithm 1.

In the forward pass, at first, the neural network, $\mathbf{Net}(\overline{\mathbf{\Phi}}(t), \boldsymbol{\theta}, t)$, approximates the chemical and heat release source terms corresponding to the normalized species mass fractions and temperature. Then, the neural network is integrated with an embedded ODE solver to yield the predicted integration values followed by loss function evaluation. The loss function is defined as MAE between the values predicted by the neural network and those obtained by direct integration for all data points in a training batch. After the completion of a forward pass, during the backward propagation, the derivatives of the loss function are backpropagated through the ODE solver and neural network to the input values as described by Eq. (13). Subsequently, the neural network weights are adjusted to minimize the loss function using the adaptive moment estimation (Adam) optimization technique for gradient descent. The number of epochs is set to 500. The learning rate is

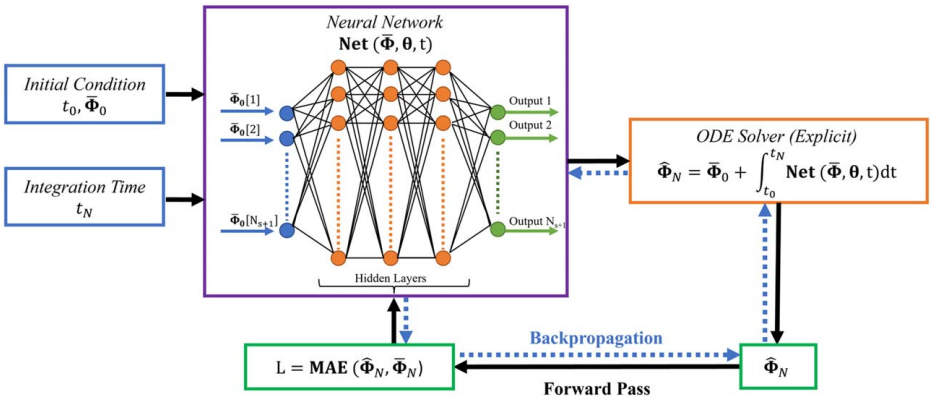

Fig. 2 Graphical representation of architecture and training workflow of an NODE chemical kinetics integrator. $\overline{\Phi}_N$ and $\hat{\Phi}_N$ represent the true and predicted solutions, respectively.

automatically reduced from 10^{-2} to 10^{-4} by utilizing an annealing method that tracks the error and slows down the learning rate when the error reaches a plateau. At the end of the training, the MAE converged to less than 3%. Figure 2 demonstrates the training workflow for a single training example consisting of a single initial condition $\overline{\Phi}_0$ evolving from t_0 to time t_N along a composition evolution trajectory.

Once the network is trained with the desired accuracy, it can be integrated with any ODE solver to obtain its state at any arbitrary time $t_N > t_0$. For example, with the normalized initial values $\overline{\Phi}_0$ at time t_0, integrating forward to time t_N using an ODE solver gives

$$\overline{\Phi}_N = \overline{\Phi}_0 + \int_{t_0}^{t_N} \mathbf{Net}(\overline{\Phi}, \theta, t)\, dt \tag{17}$$

Subsequently, normalized values $\overline{\Phi}_N$ can be converted to absolute composition vector Φ_N using Eq. (15).

Algorithm 1: Algorithm for NODE based chemistry integrator

Require $\mathbf{Z} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_I\}$ such that $\mathbf{X}_i = \{(t_0, \overline{\Phi}_0), (t_1, \overline{\Phi}_1), \ldots, (t_N, \overline{\Phi}_N), \forall\, i = 1, 2, \ldots, I$
$Z \in \Re^{(Ns+2)\times N\times I} \Leftarrow$ Samples of true solution at I initial conditions and corresponding N points on evolution trajectory or time instances
Require $\zeta \Leftarrow$ variable learning rate
Require $L_{threshold} \Leftarrow$ loss change threshold
Require Numerical ODE solver

```
θ_0 ⇐ initialize network parameters
for k = 1 to N_epochs ⇐ loop over number of epochs do
    for j = 1 to I ⇐ loop over number of initial conditions do
        for i = 1 to (N − 1) ⇐ loop over number of time instances do
            Φ̂_{i+1} = ODE solve( Net(Φ̄, θ, t), Φ̄_i, t_i, t_{i+1}) ⇐ get predicted solution from ODE integration
            ε_i = | Φ̂_{i+1} − Φ̄_{i+1}| ⇐ get MAE loss for current time instance
        end for
        L = 1/(N − 1) Σ_{i=1}^{N−1} ε_i ⇐ get mean loss for trajectory
        ∂L ⇐ get loss derivatives from backpropagation
        θ_i = Adam optimizer (θ_{i−1}, ζ, ∂L) ⇐ update network weight based on optimization algorithm
    end for
    L̄ = 1/I Σ_{j=1}^{I} L ⇐ get mean loss for all initial conditions
    if (L̄_j − L̄_{j−1}) < L_threshold then
        ζ ⇐ get new learning rate
    end if
end for
θ ⇐ Get optimized NODE network
```

3 Results and Discussion

3.1 Constant Pressure Homogeneous Hydrogen-Air Reactor. A proof-of-concept of the NODE chemistry integration framework is demonstrated by simulating hydrogen-air autoignition in a constant pressure homogeneous reactor for a range of initial conditions and comparing the NODE network results with those obtained by direct integration of DKM. For each initial condition corresponding to equivalence ratio (φ_0) and temperature (T_0), the solution is obtained by integrating from $t_0 = 0$ to $t_N = 10^{-2}$ s. Table 1 summarizes the initial φ_0 (rows) and T_0 (columns) for the 75 cases considered for analysis in this section. The identification (ID) number for these cases is assigned as ID = (column − 1) × 15 + row; for example, case ID = 18 corresponds to the third row ($\varphi_0 = 0.64$) and the second column ($T_0 = 1050$ K). To examine the accuracy of the solution at all time-steps, the integration is performed in a step-wise fashion; at each integration time-step the solution is obtained and its accuracy is evaluated before feeding it to the next time-step as the initial condition. Direct integration of DKM is performed with implicit multi-step variable-order (1–5) BDF solvers with an absolute error tolerance of $\eta_a = 10^{-8}$ for species mass fraction and 10^{-5} for temperature, along with relative error tolerance of $\eta_r = 10^{-6}$. The larger temperature η_a is because temperature values are of 3 − 4 orders of magnitude larger than those of species mass fractions. The NODE network is integrated with an explicit ODE solver based on the Runge–Kutta method of order 5(4) (RK45) with an absolute

Table 1 Initial conditions, φ_0 and T_0, of a constant pressure homogeneous H_2-air reactor corresponding to simulation IDs 1–75

	T_0				
φ_0	1000 K	1050 K	1100 K	1150 K	1200 K
0.50	1	16	31	46	61
0.57	2	17	32	47	62
0.64	3	18	33	48	63
0.71	4	19	34	49	64
0.79	5	20	35	50	65
0.86	6	21	36	51	66
0.93	7	22	37	52	67
1.00	8	23	38	53	68
1.07	9	24	39	54	69
1.14	10	25	40	55	70
1.21	11	26	41	56	71
1.29	12	27	42	57	72
1.36	13	28	43	58	73
1.43	14	29	44	59	74
1.50	15	30	45	60	75

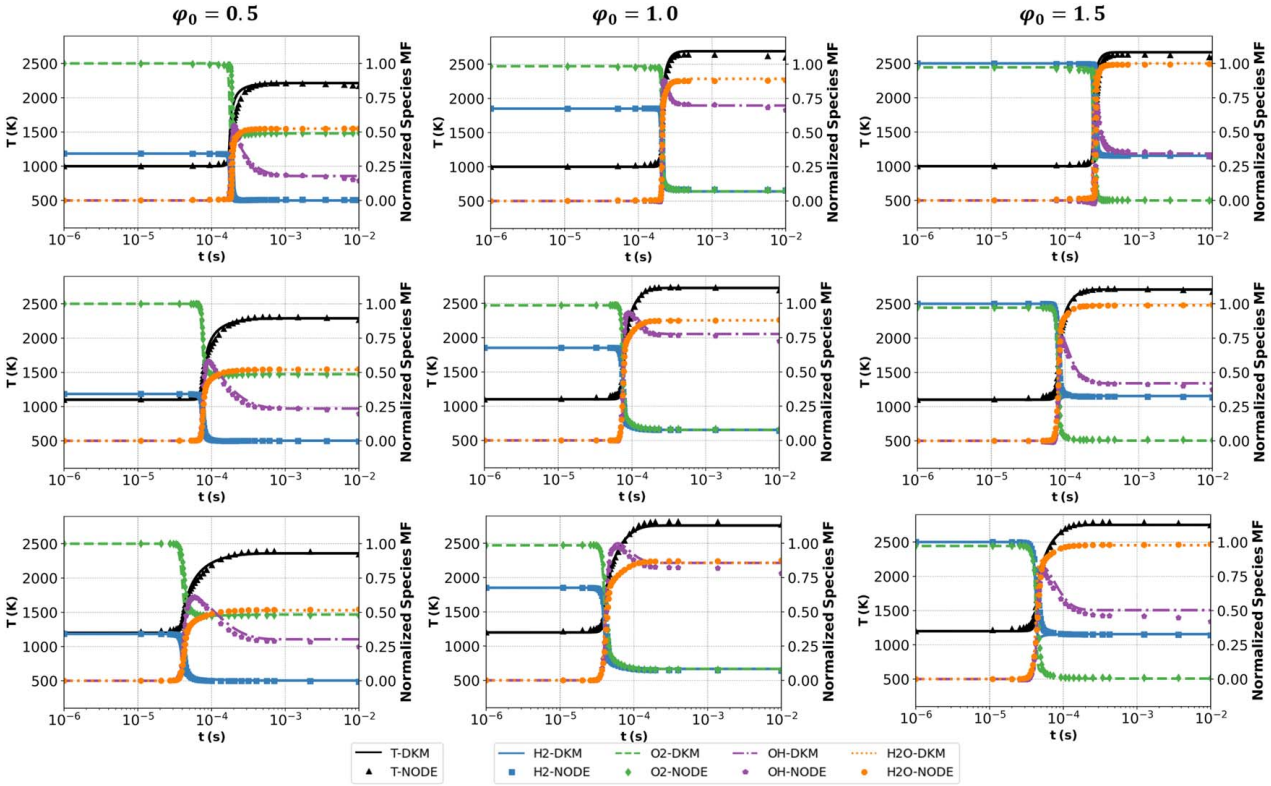

Fig. 3 Comparison of NODE network and DKM for temperature (left *y*-axis) and species MF (right *y*-axis) evolution in constant pressure homogeneous H_2-air reactor for $\varphi_0 = 0.5$, 1.0, and 1.5 (columns) at $T_0 = 1000$, 1100, and 1200 K (rows)

error tolerance of $\eta_a = 10^{-5}$, and relative error tolerance of $\eta_r = 10^{-6}$. The purpose of selecting the RK45 solver is two-fold: first, to demonstrate the ability of the NODE network to operate with non-stiff ODE solvers, and second, to highlight the flexibility of the NODE network to work with different ODE solvers than those utilized for its training. The NODE network represents the source term for normalized species and temperature according to Eq. (15); hence, the η_a value for the NODE network integration is scaled accordingly to incorporate the effect of normalization. Figure 3 shows comparisons of ignition curves obtained from integration of DKM and NODE network for gas mixtures at $T_0 =$ 1000 K, 1100 K, 1200 K and $\varphi_0 = 0.5$, 1.0, 1.5. Each subplot in Fig. 3 depicts the evolution of the temperature and MF of several species: fuel (H_2), oxidizer (O_2), product (H_2O), and OH radical —OH is a highly reactive species and can be considered as a marker of ignition in combustion applications. The NODE network satisfactorily captures the monotonic increase in temperature and H_2O as well as the non-monotonic behavior of OH. The complete ignition process including depletion of oxidizer and fuel is well captured by the NODE network. The minute discrepancy in equilibrium OH values for a few cases is attributed to the normalization of species values (Eq. (15))—the difference ($\Phi_{OH,max} - \Phi_{OH,min}$) is of order 10^{-2}, which amplifies the modest difference in absolute values of OH mass fraction to larger normalized values. Similar performance is obtained for all 75 simulations with different initial conditions (Table 1), proving the generalizability of the NODE network for a wide range of equivalence ratio and temperature values. The MAE in $\mathbf{\Phi}$ for all cases was observed to be less than 5%, signifying that the testing error is similar to the training error and that the NODE network is not overfitting.

To further investigate the characteristics of the NODE network in capturing the dynamics of chemical kinetics, Fig. 4 shows the ignition delay times as a function of φ_0 at different initial T_0 values. The ignition delay times under various conditions are well predicted by the NODE network. For the quantification of errors, the mean relative error (ε_r) of a quantity ϕ is defined as

$$\varepsilon_{r,\phi} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\phi_{i,\mathrm{DKM}} - \phi_{i,\mathrm{NODE}}}{\phi_{i,\mathrm{DKM}}}\right| \tag{18}$$

where, $\phi_{i,\mathrm{DKM}}$ and $\phi_{i,\mathrm{NODE}}$ denote the value of ϕ obtained by integrating DKM and NODE network, respectively, for the case ID i and $N = 75$ is the number of cases studies (listed in Table 1). Thus, mean relative error in ignition delay time and equilibrium temperature are represented by $\varepsilon_{r,\tau_{ign}}$ and $\varepsilon_{r,T_{eq}}$, respectively. For all simulations in Fig. 4, $\varepsilon_{r,\tau_{ign}}$ is observed to be 3.4%. A similar performance was established while assessing the accuracy of the NODE network for predicting equilibrium temperatures with $\varepsilon_{r,T_{eq}} = 1.0\%$. Overall, the NODE network provides good accuracy in predicting combustion characteristics for a range of operating conditions compared to DKM.

To assess the computational performance of the NODE network, the CPU time and the number of function evaluations (nfeval) required by the ODE solver in each simulation in Table 1 are

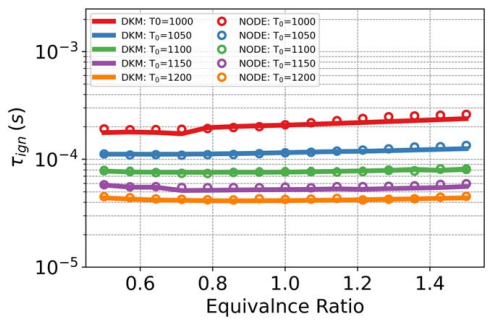

Fig. 4 Comparison of ignition delay times (τ_{ign}) at different initial temperatures (T_0) as a function of equivalence ratio (φ_0) predicted by direct integration of DKM and NODE network for a constant pressure homogeneous H_2-air reactor

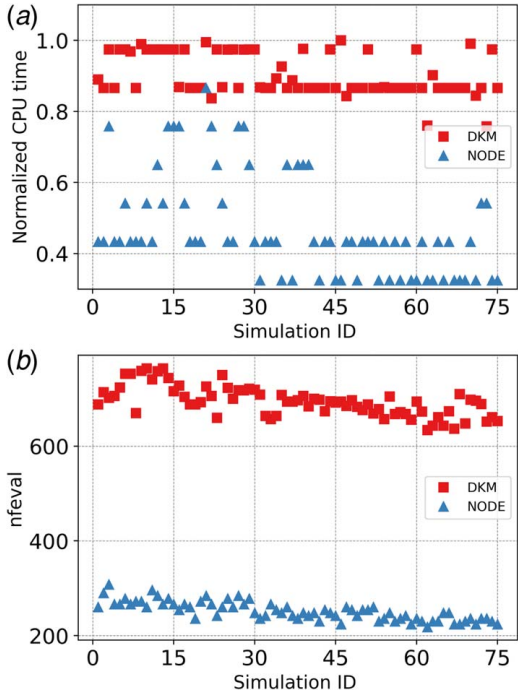

Fig. 5 (*a*) CPU time and (*b*) nfeval for each simulation in Table 1, obtained using direct integration of DKM (solver: BDF, $\eta_a = 10^{-8}$ for species mass fraction and 10^{-5} for temperature, $\eta_r = 10^{-6}$), and NODE network (solver: RK45, $\eta_a = 10^{-5}$, $\eta_r = 10^{-6}$)

recorded in Figs. 5(*a*) and 5(*b*). The CPU times are normalized with respect to the maximum CPU time for the plot. The latter indicated the number of times the source term is evaluated by the ODE solver. This number depends on the ODE solution method as well as the integration sub-interval size, which is an indication of the numerical stiffness of the system. It is evident that the NODE network (using RK45 solver) consistently outperforms DKM (using BDF solver) in terms of both CPU time and nfeval. The network reduces the function evaluation requirement of the solver by 64%. A total of 48% speed up in CPU time is achieved with the NODE network compared to DKM. It can be inferred from the observations that a single function evaluation in NODE is computationally slightly more expensive than in DKM. However, this expense is outweighed by the NODE network reduced nfeval and ODE solver overhead costs. As a result, the overall computational cost of the NODE network is lower than that of DKM. This experiment highlights the NODE network ability to operate with non-stiff ODE solvers due to its reduced stiffness. The reduced stiffness decreases the overall computational demands by effectively reducing nfeval and ODE solver overhead costs, despite that the NODE network is slightly more costly per function evaluation. The computational performance of the NODE network and DKM are further compared with the same solvers as discussed in Sec. 3.2.

3.2 Performance Assessment of NODE Network for Varied Error Tolerances and Solvers. The performance of adaptive step size ODE solvers, such as BDF and RK45, are influenced by the relative η_r and absolute η_a error tolerances. The former controls the error in the solution relative to dependent variable values. The latter provides a threshold below which the error is disregarded. This threshold determines the accuracy when the solution approaches zero. To study the effects of these solver tolerances on the accuracy and efficiency of the NODE network, the total CPU time and the mean relative errors $\varepsilon_{r,\tau_{ign}}$ and $\varepsilon_{r,T_{eq}}$ are recorded for various tolerances as shown in Fig. 6. The total CPU times are normalized with respect to the maximum CPU time for each subplot. We choose direct integration of DKM with $\eta_a = 10^{-8}$ for species mass fractions and $\eta_a = 10^{-5}$ for temperature, along with $\eta_r = 10^{-6}$ as a base case for mean error calculation using Eq. (18). Integration of NODE network is performed with RK45 for various η_a and η_r values as shown in Fig. 6. Each subplot on the top row depicts the performance of the NODE network for fixed η_a and varying η_r values. It is evident that increasing η_r reduces the CPU time. This is attributed to the fact that reducing η_r increases the number of integration sub-intervals and hence, increases nfeval by the ODE solver. The CPU time reduction with η_r is more significant at lower η_a values, where η_r is more influential in controlling the overall accuracy of the solution. For all the three cases, $\eta_a = 10^{-6}$, 10^{-5}, 10^{-4}, the variation in $\varepsilon_{r,T_{eq}}$ is minimal but $\varepsilon_{r,\tau_{ign}}$ increases with η_r. For each η_a, the $\varepsilon_{r,\tau_{ign}}$ slightly increases for $\eta_r = 10^{-7}$ to 10^{-5} and abruptly jumps for $\eta_r = 10^{-4}$ indicating that $\eta_r = 10^{-5}$ is an optimal choice to reduce CPU time without much sacrificing the accuracy. Similarly, each subplot in the bottom row depicts the performance of the NODE network for fixed η_r and varying η_a. For all three η_r values, it is observed that increasing

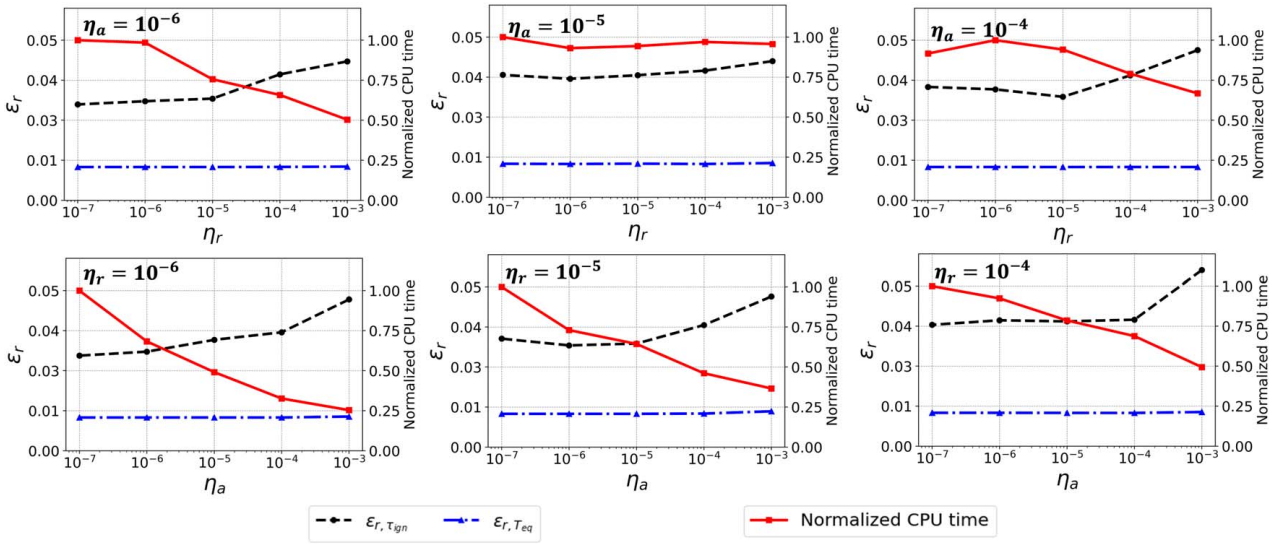

Fig. 6 The effect of varying error tolerances η_a and η_r on the NODE network prediction errors in ignition delay and equilibrium temperature (left *y*-axis) and normalized CPU time (right *y*-axis) for constant pressure homogeneous H_2-air reactor

η_a also reduces the CPU time as it decreases the nfeval needed by the ODE solver to predict near-zero solutions with lower accuracy requirements. The CPU time reduction with η_a is more pronounced at low η_r values at which the absolute tolerance is a more determining factor in controlling the integration step size. The speedup attained with varying η_a is steeper than that when η_r is changed. This suggests that predicting near-zero components of the solution with higher precision contribute more to the overall computational demand of the calculation. Regarding the accuracy, for all the three cases $\eta_r = 10^{-6}$, 10^{-5}, 10^{-4}, the variation in $\varepsilon_{r,T_{eq}}$ is minimal but $\varepsilon_{r,\tau_{ign}}$ changes significantly. For each η_r, the $\varepsilon_{r,\tau_{ign}}$ remains primarily steady until $\eta_a = 10^{-5}$ and later increases with increase in η_a. Under these observations, $\eta_a = 10^{-5}$ seems to be an optimal choice to reduce CPU time without significantly sacrificing accuracy.

We next study the performance of the NODE network when integrated with different ODE solvers using the optimal tolerances $\eta_a = 10^{-5}$ and $\eta_r = 10^{-5}$. To assess the computational performance, the CPU time and the nfeval required by each solver for each simulation in Table 1 are shown in Figs. 7(*a*) and 7(*b*) and compared with the base case. The CPU times are normalized with respect to the maximum CPU time of the plot. As expected, for specified error tolerances, explicit solver RK45 requires the least nfeval, followed by LSODA which switches between implicit BDF and explicit Adams methods during the run-time, thus reducing the number of function and Jacobian evaluations compared to fully implicit BDF solver. For all cases, it is evident that the NODE network consistently performs better than the direct integration of DKM for the same case in terms of nfeval with any given ODE solver. The consistent reduction in the number of function evaluations despite the ODE solver choice indicates that the NODE network effectively reduces the stiffness of the ODEs. The trend in CPU times is however slightly different, especially for the stiff solver, due to the competition between the cost of function evaluations and the number of evaluations required. As discussed in the previous section, the NODE network function evaluations are more costly than those of DKM.

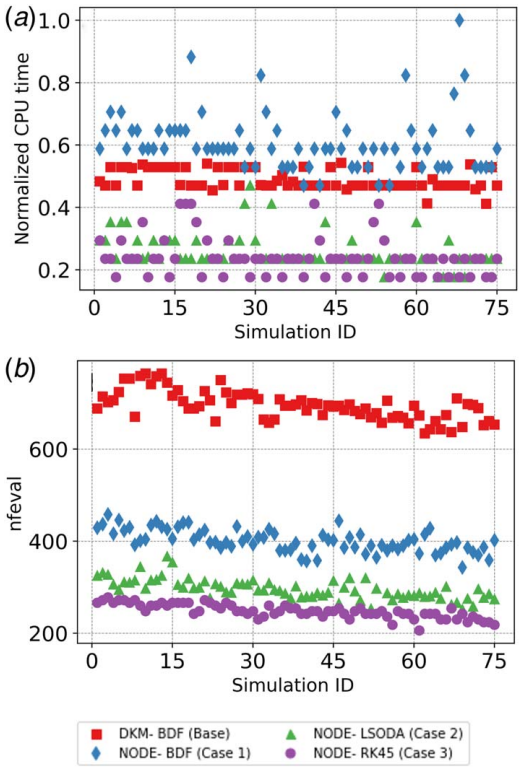

Fig. 7 (*a*) CPU time and (*b*) nfeval for each simulation in Table 1, obtained using direct integration of DKM (base case), and NODE network (Cases 1, 2, and 3). Error tolerances and $\varepsilon_{r,\tau_{ign}}$ corresponding to these cases are outlined in Table 2.

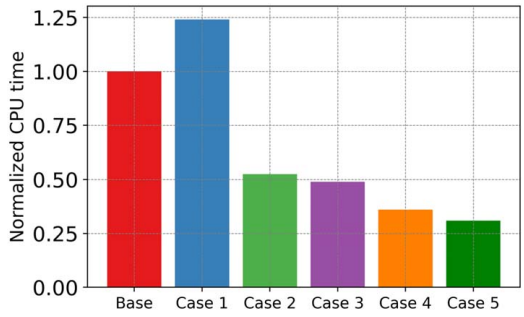

Fig. 8 Total CPU time comparison for 75 simulations listed in Table 1 for different ODE solvers. Error tolerances and $\varepsilon_{r,\tau_{ign}}$ corresponding to these cases are outlined in Table 2.

When using the BDF solver with the NODE network, the cost of function evaluations outweighs the reduced nfeval and as a result, the CPU time increases compared to the base case. On the other hand, while using LSODA and RK45 solvers with the NODE network, the nfeval is significantly low and CPU time always remains lower than the base case.

To summarize, the total CPU time comparison for all 75 simulations listed in Table 1 for different ODE solvers is displayed in Fig. 8. Summary of cases with respective tolerances and $\varepsilon_{r,\tau_{ign}}$ is presented in Table 2. As shown, Case 2 and Case 3 provide 47% and 51% reduction in CPU time compared to the base case. The performance is further improved by reducing η_a and η_r to 10^{-4} in Case 4, which shows a 64% decrease in total CPU time with less than 1% reduction in the accuracy. The solver performance is also recorded for the reduced-order solver of the explicit Runge–Kutta method of order 3(2) (RK23). The RK23 solver with similar error tolerances accelerates the solution process by 70% relative to the base case without further reduction in accuracy. These studies demonstrate the NODE network efficacy and robustness in operating with different ODE solvers and error tolerances for a complex problem of integrating stiff chemical kinetics.

Furthermore, additional gain in speed-up can be achieved by decreasing the complexity of the neural network. Neural network architecture can be optimized for the reduction in CPU time without compromising accuracy by coupling NODE with optimization algorithms, such as Bayesian, which remains a topic of further study. The computational cost for direct integration of DKM is a cubic function of a number of species [5] rendering a nonlinear increase in chemical kinetics integration cost for real fuels. On the other hand, the NODE network is a data-driven technique implying a linear increase in computational cost with respect to the number of species. Therefore, the NODE network, which is easily extendable to other commonly used complex fuels and descriptions of chemical kinetics mechanisms (detailed, skeletal, or reduced), is expected to result in a more significant reduction in the computational cost.

In light of the findings presented in this investigation, it is observed that NODE has the capability to effectively reduce the computational cost associated with chemical kinetics in combustion simulations. An important consideration in this regard involves addressing the coupling of the chemical kinetics NODE network with turbulent mixing to enable the prediction of more realistic

Table 2 Performance of different solvers and error tolerances

Case	Chemistry	Solver	η_a	η_r	$\varepsilon_{r,\tau_{ign}}$
Base	DKM	BDF	10^{-8} (10^{-5} for T)	10^{-6}	—
Case 1	NODE	BDF	10^{-5}	10^{-5}	2.95%
Case 2	NODE	LSODA	10^{-5}	10^{-5}	2.93%
Case 3	NODE	RK45	10^{-5}	10^{-5}	3.23%
Case 4	NODE	RK45	10^{-4}	10^{-4}	3.95%
Case 5	NODE	RK23	10^{-4}	10^{-4}	3.16%

cases. To this end, we have carried out an evaluation of the performance of the chemical kinetics NODE with the inclusion of mixing via a zero-dimensional pairwise mixing stirred reactor (PMSR), as detailed in Ref. [38]. Assessments based on PMSR have demonstrated that in presence of mixing, NODE can achieve even higher computational time speedup with comparable accuracy compared to direct integration of detailed kinetics, thus highlighting its potential in accelerating computations in turbulent combustion applications.

4 Conclusion

In the present study, a chemical kinetics integration methodology based on neural ordinary differential equations (NODE) is introduced. A systematic approach is presented for training data generation, pre-processing, and training of the NODE network. A proof-of-concept is demonstrated with the constant pressure homogeneous hydrogen-air reactor. The hydrogen-air chemistry is described by a finite-rate mechanism involving 9 chemical species and 21 reaction steps. Results show that the NODE simulations accurately capture the evolution of the reacting system as well as the ignition delay times for a span of equivalence ratio ranging from 0.5 to 1.5 and initial temperature from 1000 K to 1200 K. The sensitivity of the NODE with respect to the ordinary differential equation (ODE) solver error tolerances is studied and the optimal tolerance values yielding the minimum CPU time with the desired accuracy are determined. With the choice of optimal error tolerances and ODE solver, the robust NODE network in this study offers up to 70% reduction in CPU time compared to direct integration of the detailed kinetics mechanism. Chemical kinetics typically presents highly stiff ODEs requiring stiff ODE solvers to obtain the solution. An important benefit offered by the NODE network is its ability to effectively reduce the numerical stiffness and thus, enabling the use of less sophisticated ODE solvers, including the explicit ones, to solve the ODE system. This results in a lower number of function evaluations and the ODE solver overhead costs, leading to lower CPU times required to integrate the system. Therefore, the NODE network can provide a more affordable and accurate way to handle chemical kinetics in turbulent flame simulations which is a subject of future studies.

Acknowledgment

This study is supported in part by the Office of the Vice President for Research at the University of Connecticut through the Research Excellence Program.

Conflicts of Interest

There are no conflicts of interest. This article does not include research in which human participants were involved. Informed consent not applicable. This article does not include any research in which animal participants were involved.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

References

[1] Singer, M., and Pope, S., 2004, "Exploiting ISAT to Solve the Reaction-Diffusion Equation," Combust. Theory Modell., **8**(2), pp. 361–383

[2] Singer, M., Pope, S., and Najm, H., 2006, "Operator-Splitting With ISAT to Model Reacting Flow With Detailed Chemistry," Combust. Theory Modell., **10**(2), pp. 199–217.

[3] Yang, B., and Pope, S., 1998, "An Investigation of the Accuracy of Manifold Methods and Splitting Schemes in the Computational Implementation of Combustion Chemistry," Combust. Flame, **112**(1–2), pp. 16–32.

[4] Pope, S. B., 1997, "Computationally Efficient Implementation of Combustion Chemistry Using In Situ Adaptive Tabulation," Combust. Theory Modell., **1**(1), pp. 41–63.

[5] Lu, T., and Law, C. K., 2009, "Toward Accommodating Realistic Fuel Chemistry in Large-Scale Computations," Prog. Energy Combust. Sci., **35**(2), pp. 192–215.

[6] Lu, T., and Law, C. K., 2005, "A Directed Relation Graph Method for Mechanism Reduction," Proc. Combust. Inst., **30**(1), pp. 1333–1341.

[7] Vajda, S., Valko, P., and Turanyi, T., 1985, "Principal Component Analysis of Kinetic Models," Int. J. Chem. Kinet., **17**(1), pp. 55–81.

[8] Lu, T., Ju, Y., and Law, C. K., 2001, "Complex CSP for Chemistry Reduction and Analysis," Combust. Flame, **126**(1–2), pp. 1445–1455.

[9] Lu, T., and Law, C. K., 2006, "Systematic Approach to Obtain Analytic Solutions of Quasi Steady State Species in Reduced Mechanisms," J. Phys. Chem. A, **110**(49), pp. 13202–13208.

[10] Rein, M., 1992, "The Partial-Equilibrium Approximation in Reacting Flows," Phys. Fluids A, **4**(5), pp. 873–886.

[11] Law, R., Metghalchi, M., and Keck, J. C., 1989, "Rate-Controlled Constrained Equilibrium Calculations of Ignition Delay Times in Hydrogen-Oxygen Mixtures," Proc. Combust. Inst., **22**(1), pp. 1705–1713.

[12] Keck, J. C., 1990, "Rate-Controlled Constrained-Equilibrium Theory of Chemical Reactions in Complex Systems," Prog. Energy Combust. Sci., **16**(2), pp. 125–154.

[13] Hamiroune, D., Bishnu, P., Metghalchi, M., and Keck, J. C., 1998, "Rate-Controlled Constrained-Equilibrium Method Using Constraint Potentials," Combust. Theory Modell., **2**(1), pp. 81–94.

[14] Janbozorgi, M., Ugarte, S., Metghalchi, H., and Keck, J. C., 2009, "Combustion Modeling of Mono-Carbon Fuels Using the Rate-Controlled Constrained-Equilibrium Method," Combust. Flame, **156**(10), pp. 1871–1885.

[15] Hadi, F., Janbozorgi, M., Sheikhi, M. R. H., and Metghalchi, H., 2016, "A Study of Interactions Between Mixing and Chemical Reaction Using the Rate-Controlled Constrained-Equilibrium Method," J. Non-Equilibrium Thermodyn., **41**(4), pp. 257–278.

[16] Hadi, F., Yu, G., and Metghalchi, H., 2018, *Fundamentals of Rate-Controlled Constrained-Equilibrium Method*, Springer, Singapore.

[17] Maas, U., and Pope, S. B., 1992, "Simplifying Chemical Kinetics: Intrinsic Low-Dimensional Manifolds in Composition Space," Combust. Flame, **88**(3–4), pp. 239–264.

[18] Chen, J.-Y., Kollmann, W., and Dibble, R., 1989, "PDF Modeling of Turbulent Nonpremixed Methane Jet Flames," Combust. Sci. Technol., **64**(4–6), pp. 315–346.

[19] Blasco, J. A., Fueyo, N., Dopazo, C., and Ballester, J., 1998, "Modelling the Temporal Evolution of a Reduced Combustion Chemical System with an Artificial Neural Network," Combust. Flame, **113**(1–2), pp. 38–52.

[20] Blasco, J. A., Fueyo, N., Larroya, J. C., Dopazo, C., and Chen, Y. J., 1999, "A Single-Step Time-Integrator of a Methane-Air Chemical System Using Artificial Neural Networks," Comput. Chem. Eng., **23**(9), pp. 1127–1133.

[21] Chen, J. Y., Blasco, J. A., Fueyo, N., and Dopazo, C., 2000, "An Economical Strategy for Storage of Chemical Kinetics: Fitting In Situ Adaptive Tabulation With Artificial Neural Networks," Proc. Combust. Inst., **28**(1), pp. 115–121.

[22] Ding, T., Readshaw, T., Rigopoulos, S., and Jones, W. P., 2021, "Machine Learning Tabulation of Thermochemistry in Turbulent Combustion: An Approach Based on Hybrid Flamelet/Random Data and Multiple Multilayer Perceptrons," Combust. Flame, **231**, p. 111493.

[23] Laubscher, R., and Hoffmann, J. H., 2018, "Utilization of Basic Multi-layer Perceptron Artificial Neural Networks to Resolve Turbulent Fine Structure Chemical Kinetics Applied to a CFD Model of a Methane/Air Piloted Jet Flame," J. Therm. Eng., **4**(2), pp. 1828–1846.

[24] Christo, F., Masri, A., Nebot, E., and Pope, S., 1996, "An Integrated PDF/Neural Network Approach for Simulating Turbulent Reacting Systems," Symp. (Int.) Combust., **26**(1), pp. 43–48.

[25] Franke, L. L., Chatzopoulos, A. K., and Rigopoulos, S., 2017, "Tabulation of Combustion Chemistry Via Artificial Neural Networks (ANNs): Methodology and Application to LES-PDF Simulation of Sydney Flame L," Combust. Flame, **185**, pp. 245–260.

[26] Chatzopoulos, A. K., and Rigopoulos, S., 2013, "A Chemistry Tabulation Approach Via Rate-Controlled Constrained Equilibrium (RCCE) and Artificial Neural Networks (ANNs), With Application to Turbulent Non-Premixed CH_4/H_2/N_2 Flames," Proc. Combust. Inst., **34**(1), pp. 1465–1473.

[27] Ouyang, Y., Vandewalle, L. A., Chen, L., Plehiers, P. P., Dobbelaere, M. R., Heynderickx, G. J., Marin, G. B., and Van Geem, K. M., 2022, "Speeding Up Turbulent Reactive Flow Simulation Via a Deep Artificial Neural Network: A Methodology Study," Chem. Eng. J., **429**, p. 132442.

[28] Nakazawa, R., Minamoto, Y., Inoue, N., and Tanahashi, M., 2022, "Species Reaction Rate Modelling Based on Physics-Guided Machine Learning," Combust. Flame, **235**, p. 111696.

[29] Ji, W., and Deng, S., 2021, "KiNet: A Deep Neural Network Representation of Chemical Kinetics," arXiv preprint.

[30] Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D., 2018, "Neural Ordinary Differential Equations," Adv. Neural Inf. Process. Syst., **31**, pp. 6571–6583.

[31] Owoyele, O., and Pal, P., 2022, "ChemNODE: A Neural Ordinary Differential Equations Framework for Efficient Chemical Kinetic Solvers," Energy AI, **7**, p. 100118.

[32] Ji, W., Qiu, W., Shi, Z., Pan, S., and Deng, S., 2021, "Stiff-PINN: Physics-nformed Neural Network for Stiff Chemical Kinetics," J. Phys. Chem. A., **125**(36), pp. 8098–8106.

[33] Chen, R.T.Q., Amos, B., and Nickel, M., 2021, "Learning Neural Event Functions for Ordinary Differential Equations," International Conference on Learning Representations, Vienna, Austria, May 3–7.
[34] Boivin, P., Jiménez, C., Sánchez, A., and Williams, F., 2011, "An Explicit Reduced Mechanism for H_2-Air Combustion," Proc. Combust. Inst., **33**(1), pp. 517–523.
[35] "Chemical-Kinetic Mechanisms for Combustion Applications," San Diego Mechanism Web Page, Mechanical and Aerospace Engineering (Combustion Research), University of California at San Diego.
[36] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., et al., 2020, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," Nat. Meth., **17**, pp. 261–272.
[37] Goodwin, D. G., Speth, R. L., Moffat, H. K., and Weber, B. W., 2021, "Cantera: An Object-Oriented Software Toolkit for Chemical Kinetics, Thermodynamics, and Transport Processes," https://www.cantera.org, Version 2.5.1.
[38] Bansude, S., Imani, F., and Sheikhi, R., 2023, "Performance Assessment of Chemical Kinetics Neural Ordinary Differential Equations in Pairwise Mixing Stirred Reactor," ASME Open J. Eng., **2**, p. 021008.