Incomplete to complete multiphysics forecasting: a hybrid approach for learning unknown phenomena

Tathawadekar, Nilam; Doan, Nguyen Anh Khoa; Silva, Camilo F.; Thuerey, Nils

DOI
10.1017/dce.2023.20

Publication date
2023

Document Version
Final published version

Published in
Data-Centric Engineering

Citation (APA)
Tathawadekar, N., Doan, N. A. K., Silva, C. F., & Thuerey, N. (2023). Incomplete to complete multiphysics forecasting: a hybrid approach for learning unknown phenomena. *Data-Centric Engineering*, *4*(31), e27. Article e27. https://doi.org/10.1017/dce.2023.20

Data-Centric Engineering (2023), 4: e27
doi:10.1017/dce.2023.20

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Incomplete to complete multiphysics forecasting: a hybrid approach for learning unknown phenomena

Nilam N. Tathawadekar[1], Nguyen Anh Khoa Doan[2], Camilo F. Silva[3] and Nils Thuerey[1]

[1]Department of Informatics, Technical University of Munich, Garching, Germany
[2]Faculty of Aerospace Engineering, Delft University of Technology, Delft, Netherlands
[3]Department of Mechanical Engineering, Technical University of Munich, Garching, Germany
Corresponding author: Nilam N. Tathawadekar; Email: nilam.tathawadekar@tum.de

Received: 12 April 2023; **Revised:** 26 July 2023; **Accepted:** 20 September 2023

Keywords: differentiable PDE solvers; multi-physics systems; neural physics simulations; neural network model; reactive flows

Abstract

Modeling complex dynamical systems with only partial knowledge of their physical mechanisms is a crucial problem across all scientific and engineering disciplines. Purely data-driven approaches, which only make use of an artificial neural network and data, often fail to accurately simulate the evolution of the system dynamics over a sufficiently long time and in a physically consistent manner. Therefore, we propose a hybrid approach that uses a neural network model in combination with an incomplete partial differential equations (PDEs) solver that provides known, but incomplete physical information. In this study, we demonstrate that the results obtained from the incomplete PDEs can be efficiently corrected at every time step by the proposed hybrid neural network—PDE solver model, so that the effect of the unknown physics present in the system is correctly accounted for. For validation purposes, the obtained simulations of the hybrid model are successfully compared against results coming from the complete set of PDEs describing the full physics of the considered system. We demonstrate the validity of the proposed approach on a reactive flow, an archetypal multi-physics system that combines fluid mechanics and chemistry, the latter being the physics considered unknown. Experiments are made on planar and Bunsen-type flames at various operating conditions. The hybrid neural network—PDE approach correctly models the flame evolution of the cases under study for significantly long time windows, yields improved generalization and allows for larger simulation time steps.

Impact Statement

Integrating physical models with machine learning has many applications in prediction and forecasting tasks. In this work, we analyze a hybrid framework that combines neural network models with an incomplete partial differential equation (PDE) solver to account for the effects of unknown physics present in the system. We demonstrate the applicability of this approach to complex and practical simulations of reactive flows with unknown chemistry. Our experimental evaluation demonstrates the diverse possibilities this hybrid system entails such as performing accurate long-term predictions, generalization to unseen operating conditions, robustness to incorrect PDE parameters, and enabling flame shape control. This approach for hybrid simulation methods has the potential to be adapted to more complex, multi-physics problems for parameter identification and control.

This research article was awarded Open Data and Open Materials badges for transparent practices. See the Data Availability Statement for details.

1. Introduction

Modeling and forecasting of complex physical systems described by nonlinear partial differential equations (PDEs) are central to various domains with applications ranging from weather forecasting (Kalnay, 2003), design of airplane wings (Rhie and Chow, 1983; Zafar et al., 2021), to material science (Wheeler et al., 1992). Typically, a chosen set of PDEs is solved iteratively until convergence of the solution. Modeling complex physical dynamics requires a good understanding of the underlying physical phenomena. For cases where the complete information on the physics of the system is missing, deep learning models can be employed to complete the physical description when additional data of the system is available. Deep learning methods have shown promise to account for these unknown components of the system (Um et al., 2020; Yin et al., 2021). We consider a set of PDEs with partially unknown physics represented. The corresponding PDE model for a general state ϕ is given by

$$\begin{aligned}
\frac{\partial \phi}{\partial t} &= \mathcal{P}_c\left(\phi, \frac{\partial \phi}{\partial x}, \frac{\partial^2 \phi}{\partial x^2}, \ldots\right), && \text{in } \Omega \times (0, \infty) \\
&= \mathcal{F}\left(\mathcal{P}_i\left(\phi, \frac{\partial \phi}{\partial x}, \frac{\partial^2 \phi}{\partial x^2}, \ldots\right), \mathcal{P}_u\left(\phi, \frac{\partial \phi}{\partial x}, \frac{\partial^2 \phi}{\partial x^2}, \ldots\right)\right), && \text{in } \Omega \times (0, \infty) \\
\mathcal{G}(\phi) &= 0, && \text{in } \partial\Omega \times (0, \infty) \\
\phi &= h, && \text{in } \Omega \times (0),
\end{aligned} \tag{1}$$

where $\mathcal{P}_c$ represents the physical system with complete information. $\mathcal{F}$ represents a potentially simple function that combines the known but incomplete PDE description, $\mathcal{P}_i$, and unknown physics represented by $\mathcal{P}_u$ to match the solutions of complete PDE system. We take $\mathcal{G}$ and h to be known functions appropriately defining the boundary and initial conditions respectively. Ω is the spatial domain over which we solve the PDE system and $\partial\Omega$ its boundary. The term $\mathcal{P}_u$ can take the form of closure terms, source terms, higher order coupling terms between state variables or terms resulting from a set of unknown ODEs/PDEs depending on the physical system under investigation.

A commonly targeted case is when the governing equations of the complete PDE description are computationally too expensive to solve, with turbulence modeling in computational fluid dynamics (CFD) being a good example (Chen, 1997; Pope, 2000). In CFD, a spatial filtering is performed on the original governing PDEs in the context of large eddy simulations (LES). This step introduces unclosed terms in the model equations that correspond to unrepresented physics in equation (1), due to the effects of the filtered scales. Figure 1A shows instances of normalized vorticity for isotropic decaying turbulence. The solutions of a fully resolved direct numerical simulation (DNS) could be achieved by increasing the spatial resolution of LES. This is a widely studied problem, where the use of deep learning models is currently being explored (Lapeyre et al., 2019; Kochkov et al., 2021; List et al., 2022; Stachenfeld et al., 2022).

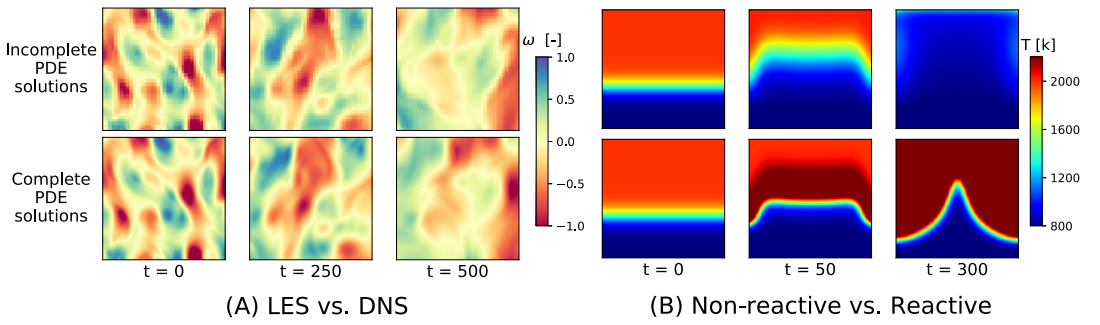

Figure 1. *(A) The normalized vorticity solutions of complete/DNS (bottom) solver can be reached by increasing the spatial resolution of the incomplete/LES (top) solver (List et al., 2022). (B) We consider the problem of the incomplete/nonreactive (top) and complete/reactive (bottom) PDE solvers which can yield fundamentally different evolutions, as shown here for a sample temperature field over time.*

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

Several of these methods train a neural network with LES as the incomplete system to model the effects of the filtered scales and obtain the solutions of the complete DNS (Kochkov et al., 2021; List et al., 2022). Instead, we consider a different, more challenging problem related to multi-physics, coupled systems containing dependent variables which describe different physical phenomena. In equation (1), $\mathcal{P}_u$ contains all the terms corresponding to different physical phenomena that are not included in the PDEs represented by $\mathcal{P}_i$. This formulation could describe fluid–structure interactions which couple fluid mechanics with structural mechanics with $\mathcal{P}_u$ representing the unknown coupling terms, or aeroacoustics problems which couple fluid dynamics and acoustics (Benra et al., 2011; Liu and Liu, 2018).

However, in this work, we will focus on reactive flow simulation as the complete PDE description, while a nonreactive flow simulation will represent the incomplete PDE basis. It collects the chemical kinetic mechanisms in the unknown physics term of equation (1). In this system, $\mathcal{P}_u$ could take the form of the source terms from unknown chemistry in the Navier–Stokes equations. A *reactive flow* refers to a fluid flow with chemical reactions occurring within a reacting fluid, such as combustion-related flows. A reacting fluid is a mixture of two or more species such as hydrocarbons, oxygen, water, carbon dioxide, and so forth, which undergo chemical reactions (Poinsot and Veynante, 2005). In contrast, a nonreactive flow refers to a fluid flow where no chemical reactions take place. Figure 1B shows a visual example of a nonreactive and reactive flow simulations which can be considered as an incomplete and complete PDE systems, respectively. Starting from the same initial condition, the influence of $\mathcal{P}_u$ in this multi-physics system leads to fundamentally different solutions as reacting fluid (shown by blue color in Figure 1B) advances through the domain without reacting for the nonreactive flow simulation or forms a Λ shaped flame with higher temperature of burned products (shown by dark red color) for the reactive flow simulation. Therefore, we are targeting a more challenging problem than those tackled in (Lapeyre et al., 2019; Kochkov et al., 2021; List et al., 2022; Stachenfeld et al., 2022), where increasing spatial resolution and/or reducing time-scales of the incomplete PDE solver does not lead to a converged full solution. Rather, the complete and incomplete PDEs produce drastically different solutions due to the unknown physics. The central learning objective is to correct this behavior and retrieve the evolution that would be obtained with the complete PDE description.

Our work expands on the combination of incomplete PDE solvers and neural networks (NNs) (Takeishi and Kalousis, 2021; Yin et al., 2021) to account for the effects of an incomplete physics model. The NN aims to complete the PDE description, where the differences in complete and incomplete PDE solutions are beyond the effects of spatial and temporal scales. We showcase that combining the trained NN model with a differentiable solver for the incomplete PDE can accurately reproduce the physical solutions of the complete, multi-physics PDE solver with stable long-term rollouts.

Reactive flow modeling has applications in numerous domains such as combustion processes in gas turbines (Lieuwen, 2012; Gruber et al., 2018), climate modeling (Jacobson, 1999; Rolnick et al., 2022), and astrophysics simulations (Gamezo et al., 2003). Resolving the Navier–Stokes equations lies at the core of these problems, where additionally the transport of different species of relevance must also be accounted for, together with their production or consumption often following complex reaction mechanisms (Poinsot and Veynante, 2005). For chemically reacting flows, the generation or consumption of multiple species via some chemical reaction is modeled using a net source term. It is a well-known fact that the incorporation of a detailed chemical kinetic mechanism in a reacting flow model can result in a stiff system of governing equations (Wanner and Hairer, 1996; Najm et al., 1998; Knio et al., 1999).

We showcase the effectiveness of our approach for different cases of planar 2D premixed methane-air flames, and the varying transient evolution of Bunsen-type flames. We show that the proposed approach can handle large domains with highly resolved flames, which are closer to the practical flame domains used in many industrial applications. Specifically, we concentrate on training a NN model to correct the spatio-temporal effects of energy and species transport source terms. We show that in addition to recovering the desired solutions, this approach overcomes inherent problems of temporal stiffness due to the complex reaction mechanism. Lastly, we show that the resulting hybrid solver provides a flexible building block for adjacent tasks. Specifically, we demonstrate this for controlling the evolution of flame shapes via continuous control. The hybrid solvers can efficiently predict the states of the reactive flows and readily

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

enable the control of the flow inlet velocities to arrive at a desired flame shape. Therefore, we employ a pre-trained hybrid solver to train a second controller network that learns to steer the flame simulation such that the observed states are matched.

This article is organized as follows. We discuss the relevant literature in Section 2. Section 3 describes the problem statement, the mathematical formulation, details of the differentiable PDE solver, and the neural network model. Discussion on data generation process is provided in Section 4. Section 5 shows the results obtained on different planar and Bunsen flame scenarios considered. Section 6 demonstrates the robustness of the proposed approach. Finally, in Section 7, we apply the proposed hybrid approach to arrive at the desired flame shape by solving an optimization problem. Section 8 concludes this article with a summary of the results and outlines future work.

2. Related work

Deep learning methods have been widely used to model the solutions of PDEs (Lagaris et al., 1998; Han et al., 2018; Long et al., 2018; Bar-Sinai et al., 2019) and in particular, the Navier–Stokes equations (Guo et al., 2016; Bhatnagar et al., 2019).

2.1. Purely data-driven models

A simple approach in modeling any physical system consists of training a deep learning model using data coming from either experiments or numerical simulation (Fukami et al., 2019; Thuerey et al., 2020). These models solely use deep learning techniques with appropriate data to make predictions and hence are called as *purely data-driven (PDD) models.* Thuerey et al. (2020) applied a convolutional neural network (CNN) to learn Reynolds-Averaged Navier–Stokes (RANS) solutions of airfoil flows. The proposed approach is very generic and applicable to a wide range of PDE boundary value problems on Cartesian grids. Stachenfeld et al. (2022) demonstrate that a generic CNN-based models may predict turbulent dynamics on coarse grids more accurately than classical numerical solvers. The proposed approach is effectively applied on a wide range of physical domains which can be represented as grids. These classical neural networks map between finite-dimensional spaces and can only learn solutions tied to a specific discretization which can be excessively limiting. Therefore, approaches based on learning operators are receiving increased attention. Lu et al. (2021) proposed a novel architecture based on fully connected neural networks called DeepONets to learn diverse linear or nonlinear explicit and implicit operators. Neural operators (Li et al., 2020B; Bhattacharya et al., 2021; Patel et al., 2021), specifically, the Fourier neural operator (FNO) of Li et al. (2020A) introduce an interesting line of work by learning mesh-free, infinite-dimensional operators with neural networks, but do not necessarily offer advantages for longer-term predictions. We compare the proposed approach with such data-driven models as baselines. PDD models can be very fast and may not suffer from the time-step stability issues associated with traditional numerical solvers. Nevertheless, as these PDD approaches lack the physical understanding of the system being modeled, they generally fail in generalizing to other operating conditions (Kim et al., 2019; Lapeyre et al., 2019). To leverage the potential of deep learning in physical simulations, it is therefore necessary to incorporate some physical information within the deep learning framework.

2.2. Physics-informed deep learning models

Deep learning models can enforce physical constraints partially through the loss function (Bar-Sinai et al., 2019; Raissi et al., 2019; Yadav et al., 2022) or changes in neural network architecture (Greydanus et al., 2019). A well-known framework called Physics-Informed Neural Network (PINN) (Raissi et al., 2019) uses neural networks as methods for solving PDEs. It minimizes the residual of the underlying governing laws by taking advantage of automatic differentiation to compute exact, mesh-free derivatives. However, these approaches struggle to enforce physical constraints such as boundary conditions or predict the strong unsteadiness and chaotic nature of flows (Dwivedi and Srinivasan, 2020; Fuks and Tchelepi, 2020).

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

PINN only learns the solution function of a single PDE instance and needs reoptimization for other instances or PDEs. To alleviate this issue, Wang et al. (2021) extended the DeepONet framework by imposing the underlying physical laws via soft penalty constraints during training. Although physics-informed DeepONet imposes PDE losses on operator learning, they are not discretization invariant. To tackle this issue, physics-informed neural operator (PINO) framework was proposed by Li et al. (2021) that uses available data and physics constraints to learn the solution operator of parametric PDEs. However, all these approaches require the explicit knowledge of the underlying PDEs to accurately train the network. For the physical systems described in equation (1) where only partial knowledge of their physical mechanism is known, these approaches may fail to converge to the solutions of the complete PDE solver. Minimizing the residual of only the known but incomplete PDEs will not lead to an accurate prediction of the solutions of a complete PDE system. Additionally, these approaches would need some further modifications to be implemented for the problem considered. Therefore, they are not considered as a baseline method for the present work.

2.3. Neural networks with differentiable PDE solvers

In recent years, the development of differentiable PDE solvers has led to an interesting line of research. Thuerey et al. (2021) have developed an open-source physics simulation toolkit called *PhiFlow* for optimization and machine-learning applications. *SU2* (Economon et al., 2016) is an open-source collection of software tools for analyzing PDEs and PDE-constrained optimization problems on unstructured meshes with state-of-the-art numerical methods. *SciML* (Rackauckas et al., 2020) is a collection of tools for solving equations and modeling systems. These tools provide differentiable functions for physical simulations, which enable close integration with deep learning frameworks by leveraging their automatic differentiation functionality. Hybrid approaches that combine machine learning techniques with numerical PDE solvers (Wang et al., 2020; Illarramendi et al., 2022), have attracted a significant amount of interest due to their capabilities for generalization (Chen et al., 2018). In this context, neural networks are typically used to model or replace a part of the conventional PDE solver to improve aspects of the solving process. For example, Tompson et al. (2017), Ajuria Illarramendi et al. (2020), and Özbay et al. (2021) proposed a convolutional neural network-based approach to solve the Poisson equation in CFD simulation. In recent years, a number of deep learning-based models have been introduced to accurately model turbulent flows (Pathak et al., 2020; Dresdner et al., 2022; Stachenfeld et al., 2022). Belbute-Peres et al. (2020) and Um et al. (2020) showed the advantages of training neural networks with differentiable physics to correct the numerical errors that arise in the discretization of PDEs. Sirignano et al. (2020) introduced a deep learning PDE augmentation method. Large eddy simulation of turbulence is augmented with a deep neural network to model unresolved physics to obtain direct numerical simulation solutions. Kochkov et al. (2021) correct for closure error by integrating a neural network with a differentiable CFD simulator. These approaches demonstrate the capabilities of neural networks to correct errors in a fast, under-resolved simulation. The unresolved physics in turbulence modeling is attributed to spatial filtering. Given additional compute resources, one can arrive at accurate solutions by running the known PDEs at higher resolution. Note that in this work we investigate a more challenging case of unknown physics in multi-physics system, where increasing the spatial or temporal resolution of the known but incomplete PDEs will not lead to the accurate solution of the complete PDEs.

Similar to the goals of our work, Takeishi and Kalousis (2021) and Yin et al. (2021) have introduced frameworks of augmenting incomplete physical dynamics with neural network models. These approaches demonstrate the applicability on ODE/PDE systems, which are weakly nonlinear and the unknown dynamics are linear combinations of the underlying flow fields. We expand on these works to explore the significantly more challenging scenario of reactive flows. These are characterized by a multi-physics system with nonlinear advective terms which models the transport of a flow state by the velocity of the flow, and strongly nonlinear dynamics, described by exponential source terms that exhibit nonlinear combinations of the flow fields. Compared to the work by Yin et al. (2021), we do not use an additive

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

approach to learn the solutions of the complete PDE system. Instead, we use the output of an incomplete PDE solver as an input to the neural network model to correct for the effects of unknown terms.

3. Methodology

We consider two different sets of PDEs with their associated numerical solver, which we denote as the *incomplete* PDE $\mathcal{P}_i$ and the *complete* PDE $\mathcal{P}_c$. By evaluating $\mathcal{P}_c$ on an input state ϕ_t at time t, we can compute the points of the phase space sequences; $\phi_{t+\Delta t} = \mathcal{P}_c(\phi_t)$. Without loss of generality, we assume a fixed time-step Δt and denote a state $\phi_{t+\Delta t}$ at next time instance as ϕ_{t+1}. Let $\mathcal{X}$ be a Banach space of functions taking values in a spatial domain $\Omega \subset \mathbb{R}^2$. Furthermore, let $C^\dagger : \mathcal{X} \to \mathcal{X}$ be a nonlinear map from ϕ_t to ϕ_{t+1}, defined over a finite time interval $[0, T]$ and an input flow state ϕ. The learning objective is to find the best possible correction function

$$C(\phi;\theta) : \mathcal{X} \to \mathcal{X}, \quad \theta \in \Theta \tag{2}$$

for some finite-dimensional parameter space Θ by choosing $\theta^\dagger$ such that $C(.;\theta^\dagger) \approx C^\dagger$. The neural network models can be employed to learn such mapping as shown in Figure 2. The model parameters θ are estimated from the complete PDE solution trajectories $(\phi_0, \phi_1, .., \phi_T)$. The learned predictions obtained after repeatedly applying the corrector C and invoking $\mathcal{P}_i$ are denoted by $(\tilde{\phi}_0, \tilde{\phi}_1, .., \tilde{\phi}_T)$.

3.1. PDEs for reactive flows

In this section, we present the basics of reactive flow simulation and underlying conservation equations. A reactive flow involves multiple species reacting through one or more chemical reactions. Different species, such as hydrocarbons (fuel), oxygen (oxidizer), water, carbon dioxide (products), are characterized through their mass fraction Y_k. The primary variables for two-dimensional (2-D) reacting flow involve density (ρ), 2-D velocity field (u), temperature (T), and the mass fraction Y_k of the N reacting species.

In a premixed combustor, fuel (Y_F) and oxidizer (Y_O) are mixed before they enter the combustion chamber. The computation of premixed flames with complex chemistry is possible but we consider a simplified approach. We assume that chemistry proceeds only through one irreversible reaction, that is, one-step chemistry. If v'_F and v'_O are the coefficients corresponding to fuel and oxidizer when considering a one-step reaction of type $v'_F F + v'_O O \to$ products, the mass fractions of fuel and oxidizer correspond to stoichiometric conditions. It is defined as,

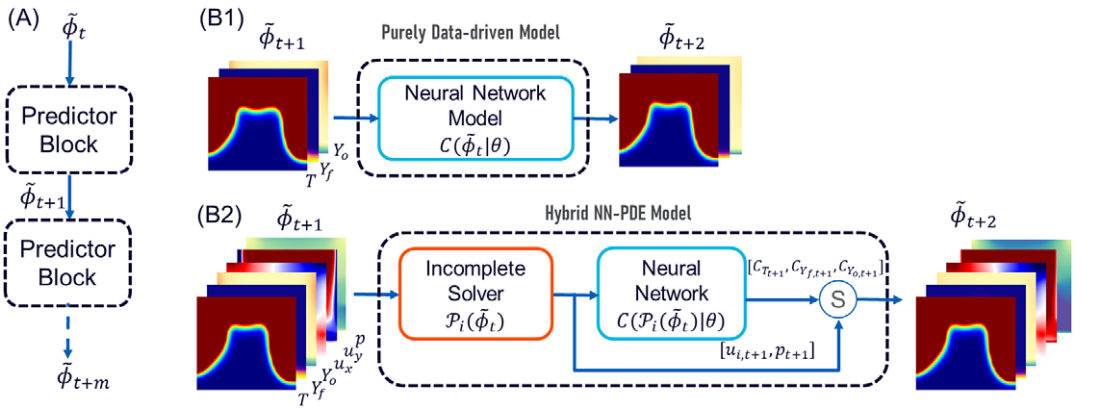

Figure 2. *(A) Multi-step training framework helps to learn the dynamics of complete PDE solver over longer rollouts. (B1) Details of the input flow state and predictor block used in a purely data-driven approach and (B2) the hybrid NN-PDE approach, where S denotes the concatenation of different fields to obtain the complete flow state $\tilde{\phi}$ at next time step.*

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

$$\left(\frac{Y_O}{Y_F}\right)_{st} = \frac{v'_O W_O}{v'_F W_F} = \varphi. \tag{3}$$

This ratio φ is called the mass stoichiometric ratio. W_F and W_O represent the molecular weights of the fuel and oxidizer, respectively. The equivalence ratio of a given mixture is then,

$$E = \varphi\left(\frac{Y_F}{Y_O}\right). \tag{4}$$

A common example of such reaction is $CH_4 + 2O_2 \rightarrow CO_2 + 2H_2O$, where $v'_F = 1, v'_O = 2$, $W_F = 0.016\text{kg/mol}, W_O = 0.032\text{kg/mol}$ and therefore $\varphi = 4$. The equivalence ratio is an important parameter in the design of a premixed combustion system. Rich combustion is observed for $E > 1$ (the fuel is in excess) and lean regimes are achieved when ($E < 1$) (the oxidizer is in excess) (Poinsot and Veynante, 2005). Most practical premixed combustors operate at or below stoichiometry (Lieuwen and Yang, 2005).

The physical system we investigate is a laminar premixed methane-air flame with one-step chemistry. It is governed by the following Navier–Stokes equations (Poinsot and Veynante, 2005)

$$\begin{aligned}
\frac{\partial \rho}{\partial t} + \nabla.(\rho u) &= 0 && \text{in } \Omega \times [0,T] \\
\frac{\partial}{\partial t}(\rho u) + \nabla.(\rho u \otimes u) &= -\nabla p + \nabla.\tau && \text{in } \Omega \times [0,T] \\
\rho C_p\left(\frac{\partial T}{\partial t} + \nabla.(u \otimes T)\right) &= \dot{\omega}'_L + \lambda \nabla^2 T && \text{in } \Omega \times [0,T] \\
\frac{\partial \rho Y_k}{\partial t} + \nabla.(\rho u \otimes Y_k) &= \dot{\omega}_k + \rho D_k \nabla^2 Y_k && \text{in } \Omega \times [0,T],
\end{aligned} \tag{5}$$

where τ, D_k, and λ are the strain rate tensor, the diffusion coefficient of species k, and the mixture thermal conductivity. In addition, C_p denotes the mixture specific heat capacity. $\dot{\omega}_k$ and $\dot{\omega}'_T$ are the species reaction rate and heat release rate, respectively. Boundary conditions of each flow state vary depending on the applications and are provided in detail in Section 4. $\Omega \subset \mathbb{R}^2$ represents the spatial domain.

The reaction rate $\dot{\omega}_k$ for each species is linked to the progress rate Q_1 at which the single reaction proceeds, as: $\dot{\omega}_k = W_k v_k Q_1$. The simplifications proposed by Mitani (1980) and Williams (1985) are used to model the reaction rates $\dot{\omega}_k$ for each species. The progress rate Q_1 is assumed to have the Arrhenius form and is given by,

$$\begin{aligned}
Q_1 &= B_1 T^{\beta_1}\left(\frac{\rho Y_F}{W_F}\right)^{n_F}\left(\frac{\rho Y_O}{W_O}\right)^{n_O}\exp\left(-\frac{E_a}{\text{RT}}\right). \\
\dot{\omega}_F &= v'_F W_F Q_1; \quad \dot{\omega}_O = v'_O W_O Q_1,
\end{aligned} \tag{6}$$

where $\dot{\omega}_F$ and $\dot{\omega}_O$ are the reaction rates of fuel and oxidizer, respectively. $\dot{\omega}'_T$ is the heat release due to combustion and is formulated as, $\dot{\omega}'_T = -\sum_{k=1}^{N} \Delta h^o_{f,k} \dot{\omega}_k$. The formation enthalpy $h^o_{f,k}$ is the enthalpy needed to form 1 kg of species k at the reference temperature $T_0 = 298.15\text{K}$. The formulation for $\dot{\omega}'_T$ can be further simplified as,

$$\dot{\omega}'_T = -\sum_{k=1}^{N} \Delta h^o_{f,k} W_k v_k Q_1 = -\sum_{k=1}^{N} \Delta h^o_{f,k} \frac{W_k v_k}{W_F v_F} W_F v_F Q_1 = -\sum_{k=1}^{N} \Delta h^o_{f,k} \frac{W_k v_k}{W_F v_F} \dot{\omega}_F = -Q\dot{\omega}_F. \tag{7}$$

Therefore, the heat release source term $\dot{\omega}'_T$ and the fuel source term $\dot{\omega}_F$ are linked by the Q, which is the heat of reaction per unit mass. Following Poinsot and Veynante (2005), parameters corresponding to a real-world methane-air flame are chosen as: $B_1 = 1.0810^7\text{uSI}$; $\beta_1 = 0$; $E_a = 83600\ \text{J/mole}$; $n_F = 1$; $n_O = 0.5$; $Q = 50100\ \text{kJ/kg}$; $C_p = 1450\ \text{J/(kgK)}$. Taken together, the system of equations above is a challenging scenario even for classical solvers, and due to its practical relevance likewise a highly interesting environment for deep learning methods.

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

3.2. Problem formulation

The incomplete PDE solver solves the set of equation (5) without the source terms and reaction rates $\dot{\omega}_k$ and $\dot{\omega}'_T$, while the complete PDE solver solves the full set of equation (5). The neural network model, denoted by $C(\mathcal{P}_i(\phi)|\theta)$, corrects the incomplete/nonreacting flow states $\mathcal{P}_i(\phi)$ to obtain the complete/reacting states $\mathcal{P}_c(\phi)$ as shown in Figure 2B2. The neural network is trained to model the effects of the unknown chemistry using parameters θ given an input flow state, $\phi = [u_i, p, T, Y_f, Y_o]$. As seen from equations (5)–(7), the temperature and species mass fraction fields are strongly coupled, which significantly increases the prediction problem complexity. A slight error in one of the fields will quickly propagate into the other fields, making the predictions diverge. In the following, a subscript $C_s(\circ)$ will denote that the neural network C generates the field s, for example, C_T generating the temperature field. The update can be written as, $\tilde{\phi}_{t+1} = [u_{i,t+1}, p_{t+1}, C_T(\mathcal{P}_i(\tilde{\phi}_t)|\theta), C_{Y_f}(\mathcal{P}_i(\tilde{\phi}_t)|\theta), C_{Y_o}(\mathcal{P}_i(\tilde{\phi}_t)|\theta)]$ where $\tilde{\cdot}$ indicates a corrected state. $u_{i,t+1}$ and p_{t+1} are the time-advanced velocity and pressure field, respectively, predicted using the incomplete PDE solver.

3.3. Training methodology

3.3.1. Hybrid NN-PDE approach

We employ a hybrid NN-PDE approach that augments a neural network model with a PDE solver (Um et al., 2020; Kochkov et al., 2021). In contrast to previous work, we use the incomplete PDE solver as a basis, and hence the solver does not converge to the desired solutions under refinement, as explained in Section 1. The neural network is integrated and trained in a loop with the incomplete PDE solver using stochastic gradient descent for m time steps, as shown in Figure 2A, B2. Here, the number of temporal look-ahead steps, m, is an important hyper-parameter of the training process. Higher m provides the network with longer-term feedback at training time through the gradient roll-outs. This gives the model improved feedback on how the time dynamics of the incomplete PDE solver affect the input states, and hence which corrections need to be inferred by the model. In our framework, if an incomplete PDE solver is very close to the complete PDE solver for the given data, the NN model would learn a correction which is closer to an identity. The skip connections in the used convolutional neural network architectures (Figure 3) would readily enable such an identity operation. In contrast, when the incomplete solver contributes very little, the proposed approach would start to represent a PDD approach.

3.3.2. Differentiable PDE solver

A central component of the hybrid NN-PDE model is the differentiable solver, which allows us to embed the solver for the incomplete PDE system in the training of a neural network. The differentiable solver acts as additional nontrainable layers in the network. They provide derivatives of the outputs of the simulation with respect to its inputs and parameters. Finite differences can be used to compute the gradients of the PDE solver, but they are computationally very expensive for high-dimensional PDEs. Differentiable solvers resolve this issue by solving the adjoint problem (Pontryagin, 1987) via analytic derivatives. Here,

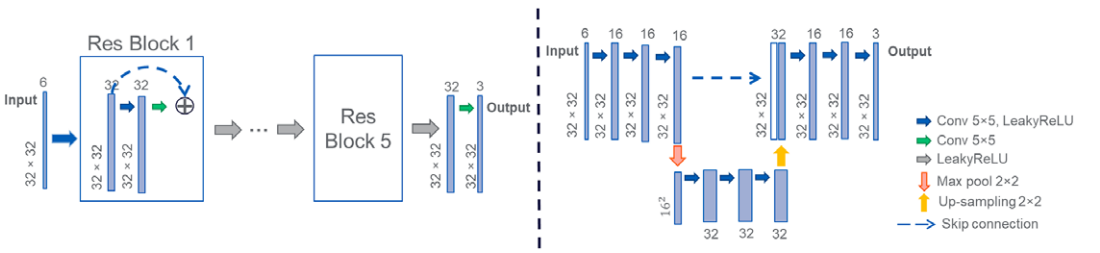

Figure 3. *Schematic of the convolutional neural networks used. Left: ResNet with 5 ResBlocks, Right: UNet32 with 2 layers.*

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

we use the differentiable PDE solver from the *phiflow* framework of Thuerey et al. (2021) in combination with *Tensorflow* to obtain the nonreactive flow solver and reactive flow solver solutions. The marker-and-cell method (Harlow and Welch, 1965) is adopted to represent temperature, pressure, density, species mass fraction fields in a centered grid, and velocities in a staggered grid. Basic differential operators such as gradient, divergence, curl, and Laplace operators are implemented in TensorFlow using basic mathematical tensor operations (Holl et al., 2020). These differential operators act on a 9-point stencil of grid points and the corresponding derivatives are straight-forward to compute. Advection is implemented with a semi-Lagrangian step, while the Poisson problem for the pressure and its gradient is solved implicitly. Efficient derivatives for all these operations are then combined via back-propagation (Werbos, 1990).

3.3.3. PDD approach

A PDD model is used as a baseline. It employs a neural network model to learn the complete flow states $\mathcal{P}_c(\phi)$ given an input flow state ϕ^S where $\phi^S = [T, Y_f, Y_o]$, as shown in Figure 2B1. The new state predicted by the trained neural network model is $\tilde{\phi}^S_{t+1} = C\left(\tilde{\phi}^S_t | \theta\right)$. For the hybrid NN-PDE as well as PDD, the neural network part of the predictor block in Figure 2 consists of a fully convolutional neural network model.

We additionally compare against the *Fourier neural operator* (FNO) of Li et al. (2020A) as an example of a state-of-the-art neural operator method. The new state predicted by the trained model is given by $\tilde{\phi}^S_{t+1} = C\left(\tilde{\phi}^S_t | \theta\right)$ where $\tilde{\phi}^S = [T, Y_f, Y_o]$ and $C(\circ)$ represents the Fourier neural operator.

3.4. Training details

All three approaches use an L_2 based loss that is evaluated for m steps as

$$\mathcal{L}(\theta) = \sum_{n=t+1}^{t+m} \sum_{\phi = \{T, Y_f, Y_o\}} \left\| \tilde{\phi}_n, \mathcal{P}_c(\phi_n) \right\|_2. \tag{8}$$

The network receives the input states as shown in Figure 2. The output of the neural network model is used to obtain the corrected state $\tilde{\phi}_{t+1}$ as specified above. We constrain the mass fraction fields Y_k to contain physical values in the range $Y_f \in [0, 0.05]$ and $Y_o \in [0, 1]$. All models are trained for 100 epochs with a batch size of 3 and a learning rate of 0.0001. We have used a small batch size of 3 due to the memory requirements as we unroll the NN-PDE framework over long timesteps. Our training procedure uses the Adam optimizer (Kingma and Ba, 2015). For all our computations, a *Nvidia Quadro RTX 8000* GPU is used. Figure 4 shows the typical training loss curve over 100 epochs.

For PDD and the hybrid NN-PDE approach, we experimented with both the ResNet (He et al., 2016) and the UNet (Ronneberger et al., 2015) architectures. Figure 3 shows the schematic of the neural network models used. We found that the ResNet performed the best for the PDD setting, while for the hybrid NN-PDE approach, the UNet performed consistently better. Hence, the following results will use a ResNet for PDD models, and a UNet for the NN-PDE hybrids. Additional details of the neural network architectures are provided in Section A.1 of the Supplementary Material as well as results comparing the UNet and ResNet architectures.

4. Numerical experiments

We consider a case of planar 2D premixed methane-air flame propagating in a quiescent mixture (***Planar-v0***) and two cases of the transient evolution of an initially planar laminar premixed flame into a Bunsen-type flame under different inlet velocity conditions (***uniform-Bunsen, nonUniform-Bunsen***). We obtain the target data by considering the source terms as defined by equations (6) and (7) with the parameters mentioned in Section 3.1.

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

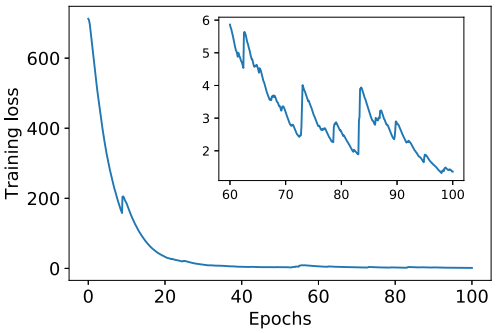

Figure 4. *Typical L_2 based training loss as defined in equation (8) over 100 epochs. The inset figure shows zoomed-in loss function curve for last 40 epochs.*

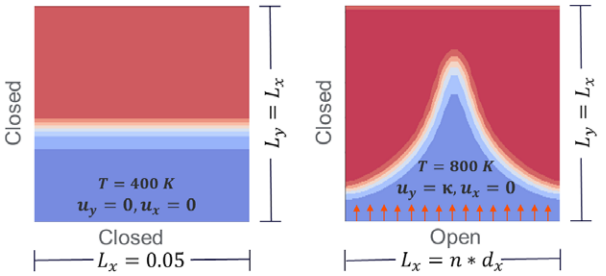

Figure 5. *Details of the boundary condition for—left: Planar flame; right: a Bunsen-type flame. n represents resolution of the domain.*

4.1. Planar-v0

For the most basic case, the planar 2D flame model setup, we consider the reacting Navier–Stokes equations described in Section 3.1 with zero inlet velocity, that is, $u_x = 0$, $u_y = 0$. We consider a square domain of size 0.05 m × 0.05 m with 32 × 32 resolution and closed boundary conditions, as shown in Figure 5. The simulation is initialized using a steep transition between a premixed methane-air mixture and burnt gases. Our training data consists of six simulations of 300 steps created by varying the equivalence ratio E. It represents the stoichiometric mixture (φ) of fuel Y_f and oxidizer Y_o mass fractions, that is, $E = \varphi \frac{Y_f}{Y_o}$ and thus fundamentally influences the dynamics of the chemical reaction. For the training data we use $E_{\text{train}} = \{1.0, 0.9, 0.8, 0.7, 0.6\}$, while the test dataset contains $E_{\text{test}} = \{0.95, 0.85, 0.75, 0.65\}$.

4.2. uniform-Bunsen

In contrast to the planar case, the premixed methane-air mixture is now fed with a constant inlet velocity. The boundary conditions upstream, at $y = 0$, are $(u_x, u_y)_{x,y=0} = (0, \kappa)$ where κ is a n-dimensional vector with constant amplitude. The target simulation contains a heat release rate term, and in this case, the initial temperature field evolves into different Λ-shaped flames at the end of the 300th time step. Training and testing datasets are created by varying the equivalence ratio and inlet velocity amplitude U: $E_{\text{train}} = \{1.0, 0.9, 0.8\}$ and $U_{\text{train}} = \{0.45, 0.4, 0.3\}$. The test dataset uses $E_{\text{test}} = \{0.95, 0.85\}$ $U_{\text{test}} = \{0.43, 0.375, 0.325\}$. The length and temperature of the flame significantly vary depending on the inlet velocity and equivalence ratio provided. Figure 6A shows the temperature field evolution of the incomplete PDE solver at different time instances for 3 different operating conditions for the uniform-Bunsen case. Figure 6B shows the corresponding target training data for the uniform-Bunsen case. All the

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

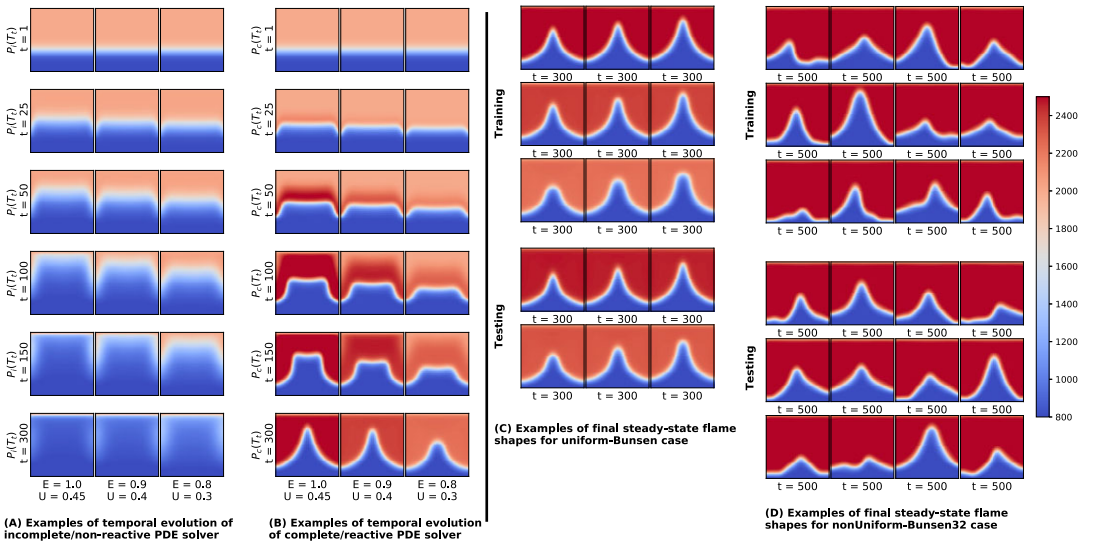

Figure 6. *Instances of temporal evolution of (A) incomplete PDE solver; (B) complete PDE solver for different operating conditions. (C) Snapshot of complete PDE solver at* $t = 300$ *for all training and testing datasets of the uniform-Bunsen case. (D) Snapshot of complete PDE solver at* $t = 500$ *for all training and testing datasets of the nonUniform-Bunsen32 case.*

simulations are run for 300 steps. It shows the difference between nonreactive and reactive flow solver simulations for different operating conditions over 300 time-steps. Figure 6C shows the last snapshot ($t = 300$) of nine training simulations and six testing datasets with operating parameters interpolated between the training data parameters for uniform-Bunsen case. These datasets are obtained by varying the equivalence ratio and the magnitude of the uniform inlet velocity. The flame temperature depends on the equivalence ratio used and the flame height depends on the inlet velocity amplitude and equivalence ratio.

4.3. nonUniform-Bunsen

As a third case, we consider the transient evolution of a premixed methane-air flame with nonuniform inlet velocity. The boundary conditions upstream, at $y = 0$, are $(u_x, u_y)_{x,y=0} = (0, \kappa)$ where κ is a n-dimensional vector whose elements are each sampled from a uniform distribution from $[0.2, 0.65]$. We experiment with two different domain sizes, 32 × 32 (***nonUniform-Bunsen32***) and 100 × 100 (***nonUniform-Bunsen100***). The larger domain size used is closer to the practical reactive flow domain utilized in CFD applications (Jaensch et al., 2017) with a highly resolved flame. These inlet velocity conditions generate complex flame shapes, which increase the difficulty of the prediction problem. We consider simulation sequences with 500 steps, as it takes longer time for the flame to reach the steady-state solution. Figure 6D showcases the snapshots of training and testing dataset at $t = 500$ for nonUniform-Bunsen32 case. Nonuniform variations in inlet velocity profile leads to different complex flame shapes. We use 12 datasets with 500 simulation steps to train the models and test it on 12 test cases shown in Figure 6D. For the nonUniform-Bunsen32 case with 32 unrolling steps ($m = 32$), training requires approximately 60 hours with approximately 1 GB of GPU memory.

To generate input and target data for training, we simulate the temperature T, mass fractions Y_f, Y_o and velocity u_x, u_y fields of all flames under study. Table 1 summarizes the boundary conditions applied for the planar and Bunsen-type flame cases discussed above. No-slip boundary conditions are used for the velocity field u_y to obtain the Λ-shaped flames.

The training and test datasets are split using different initial conditions. This ensures that varied (training) and unseen (test) data are provided. For example, in the case of Planar-v0 scenario, the training

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

Table 1. *Details of the boundary conditions used for the planar flame case and various cases of Bunsen-type flames.*

Planar flame			Bunsen-type flame		
Field	Inlet	Left and right wall	Field	Inlet	Left and right wall
T	400 K	Neumann BC	T	800 K	Neumann BC
u_y	0	—	u_y	κ	No-slip
u_x	0	—	u_x	0	Slip
p	101,325	Neumann BC	p	101,325	Neumann BC
Y_f	$\frac{1}{1+\frac{\varphi}{E}\left(1+3.76\frac{W_{N_2}}{W_{O_2}}\right)}$	Slip	Y_f	$\frac{1}{1+\frac{\varphi}{E}\left(1+3.76\frac{W_{N_2}}{W_{O_2}}\right)}$	Slip

Note. The uniform-Bunsen case is obtained with κ = constant and nonUniform-Bunsen case is obtained with $\kappa \in \mathbb{R}^n$ as discussed in Section 4.

data consists of five simulations of planar flame with different equivalence ratios, evolving over 300 simulation steps. During test time, only the initial state of the flow field (ϕ_0) and unseen boundary conditions (i.e., other equivalence ratios than those in the training datasets) are specified and the trained hybrid NN-PDE model is used to make predictions over 300 time steps. A similar setup is used for the baseline approaches. Multi-step training framework shown in Figure 2 demonstrates the training setup for the baseline and hybrid NN-PDE approaches. Continuous time-slices are used during training. As shown in Figure 2A, during training, the *Predictor Block* is unrolled for m steps, that is, the network is trained using multiple "sequences" of m consecutive snapshots (selected from the cases in the training datasets). During testing, only ϕ_0 (the initial flow state) along with unseen boundary conditions are provided and the trained predictor block is evaluated 300 times to obtain predictions from $t = 1, 2 \ldots, 300$.

5. Results

We demonstrate the capabilities of the proposed learning approach to represent the complete PDE description with the aforementioned cases of increasing difficulty. We also study its ability to generalize to unseen operating conditions such as equivalence ratios, simultaneous variations in constant inlet velocity and equivalence ratio, and nonuniform inlet velocity profiles. As baselines, we compare against a PDD approach; a neural network model with exactly the same look-ahead steps, and the Fourier neural operator of Li et al. (2020A) that likewise includes m look-ahead steps as discussed in Section 3.3. All the qualitative and quantitative evaluations shown in the article are performed on test datasets with unseen initial conditions.

5.1. Planar-v0

Table 2 compares the mean absolute percentage errors (MAPE) and mean squared errors (MSE) of temperature field for all the cases discussed in Section 4. For Planar-v0, the FNO and PDD approaches yield large errors with a MAPE of 8.27% and 6.33%, respectively. On the other hand, the hybrid NN-PDE model trained with 32 look-ahead steps reduces the error to 1.4% and thus performs significantly better than the two baselines. This behavior is visualized in Figure 7 with a 1D transverse cut of the simulation domain over 300 time-steps. The hybrid NN-PDE approach successfully captures the propagation of the flame.

In Figure 8, we also compare two important physical quantities: the flame temperature and relative displacement of the flame front, across different equivalence ratios. It can be seen that the hybrid NN-PDE model (green circles) accurately predicts the flame temperature for different test cases (solid green line). The relative displacement of the flame front is computed as $|\tilde{x}_t - \tilde{x}_0|/|x_t - x_0|$, where x_t is the position along the flame normal at time t on the 1,200K isotherm of the ground truth simulation, and $\tilde{x}$ denotes the

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

Table 2. *Mean and standard deviation of errors over all time steps of all testsets.*

		Baseline	Baseline	Hybrid	Hybrid
		FNO	PDD	NN-PDE	NN-PDE-dt
MAPE	Planar-v0	8.27 ± 5.51%	6.33 ± 3.05%	1.40 ± 0.65%	**1.21** ± 0.39%
	uniform-Bunsen	15.57 ± 8.72%	7.58 ± 3.73%	**0.72** ± 0.37%	1.11 ± 0.47%
	nonUniform-Bunsen32	12.30 ± 7.98%	12.48 ± 11.31%	**2.04** ± 1.39%	2.46 ± 1.61%
	nonUniform-Bunsen100	—	—	**3.23** ± 3.76%	4.14 ± 4.99%
MSE	Planar-v0	88,316 ± 124,394	34,491 ± 43,966	1,122 ± 1,300	**292** ± 296
	uniform-Bunsen	220,817 ± 167,024	72,563 ± 56,919	**721** ± 1,007	1,267 ± 1,622
	nonUniform-Bunsen32	184,860 ± 185,694	130,219 ± 142,094	**9,647** ± 13,903	17,569 ± 24,392
	nonUniform-Bunsen100	—	—	**23,293** ± 37,451	26,862 ± 47,639

Note. The Hybrid NN-PDE approach outperforms all other baselines considered.

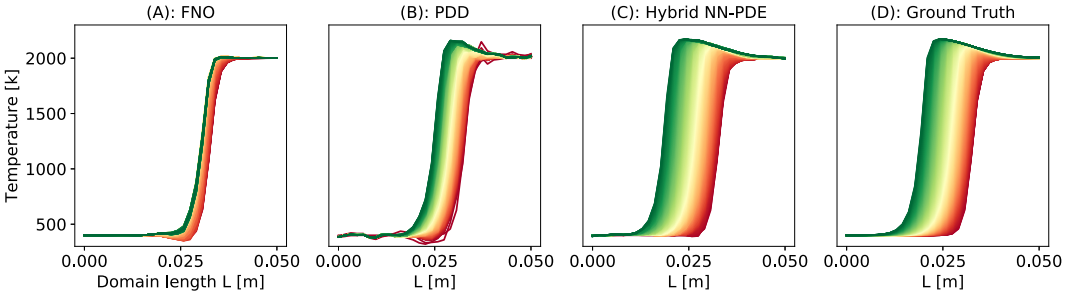

Figure 7. *1D cut of the planar flame simulation over 300 steps. The initial state is plotted in red, target state in green. Hybrid NN-PDE approach predicts physically accurate results over longer rollouts.*

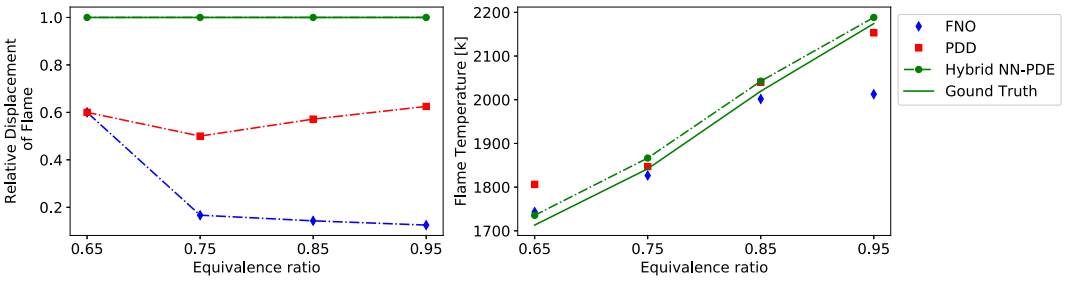

Figure 8. *Hybrid NN-PDE approach predicts physically accurate results with correct flame temperature and relative displacement of flame front across different equivalence ratios.*

predicted position. For all test cases, the hybrid approach accurately predicts the flame displacement over $t = 300$ steps, while the other approaches yield significant errors. Figure 9A shows the 2D visualization of the temperature field predictions for Planar-v0 case. As seen from the ground truth images, methane-air mixture (black color) converts into the burned products (yellow color) due to the chemical reaction at the flat flame surface (red color). The dotted, horizontal red line helps to compare the transition of the flame interface, that is, the displacement of the flame front. Due to the chemical reaction, fuel-air mixture is

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

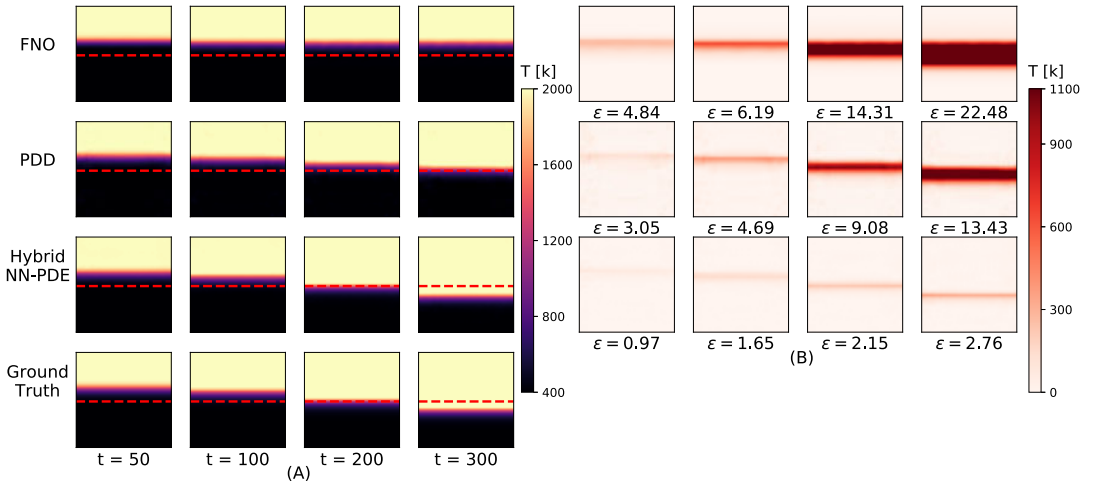

Figure 9. *Planar-v0 flame case with E = 0.95. (A) Temperature field predictions; (B) Absolute error between ground truth ($\mathcal{P}_c(T_t)$) and the output predicted—from top to bottom top—by: FNO, purely data-driven approach and hybrid NN-PDE. The numbers represent instantaneous MAPE.*

consumed and turns into burned product as the simulation progresses. The FNO approach completely fails to predict the propagation of the flame for the given test case. Its output does not show any evolution from the initial temperature profile for the given operating condition. The PDD approach fails to capture the flame front displacement correctly, thus leading to an inaccurate prediction with large errors. The hybrid NN-PDE model accurately captures this evolution of planar methane-air flame in a quiescent mixture. Figure 9B shows the instantaneous MAPE w.r.t. ground truth data for predictions shown in Figure 9A. The absolute error shown in Figure 9B exceeds 1,100 K for the FNO and PDD approaches as these do not predict the flame temperature and flame front displacement correctly. We use the upper limit of 1,100 K for colorbar to highlight the errors in the hybrid NN-PDE approach more clearly. Large errors in FNO and PDD results stem from their inability to reliably predict the flame temperature and flame front displacement.

5.2. Bunsen-type flame

For the uniform-Bunsen case, the PDD baseline with an error of 7.58% performs better than the FNO model which yields an error of 15.57%. However, a neural network model combined with an incomplete PDE solver for 32 look-ahead steps yields a significantly lower error of 0.72% as shown in Table 2. This means that the hybrid approach reduces the errors by a factor of 10 over the baselines considered. Figure 10A shows the temporal evolution of the hybrid NN-PDE approach predicted over 300 time steps for the uniform-Bunsen case. It shows the results for a test case with $U = 0.4375$ and $E = 0.95$. It predicts a symmetric flame with accurate flame height and achieves very low instantaneous MAPE of 0.73% at t=300, when compared with the ground truth data.

Next, we study a complex scenario of nonUniform-Bunsen flame with 32 × 32 resolution. The hybrid NN-PDE approach outperforms the PDD approach and FNO with an improvement of $\sim 80\%$. We also include variant of the PDD approach (PDD-5) which is trained to predict all quantities ($\phi^S = [T, Y_f, Y_o, u, p]$) of the flow state (details in Section A.2 of the Supplementary Material). The hybrid NN-PDE model achieves a low MAPE of 2.04% compared to the higher errors of 23.45% and 12.48% predicted by the PDD-5 and the PDD approach, respectively. The MSE values show this trend even more clearly. The large standard deviations of the MSE numbers indicate that the predictions made by the baseline approaches contain substantial deviations from the target values. It is important to highlight that despite using the same training data and a similar neural network architecture with same number of look-ahead steps, the hybrid approach outperforms the PDD approach due to its learned collaboration with the

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

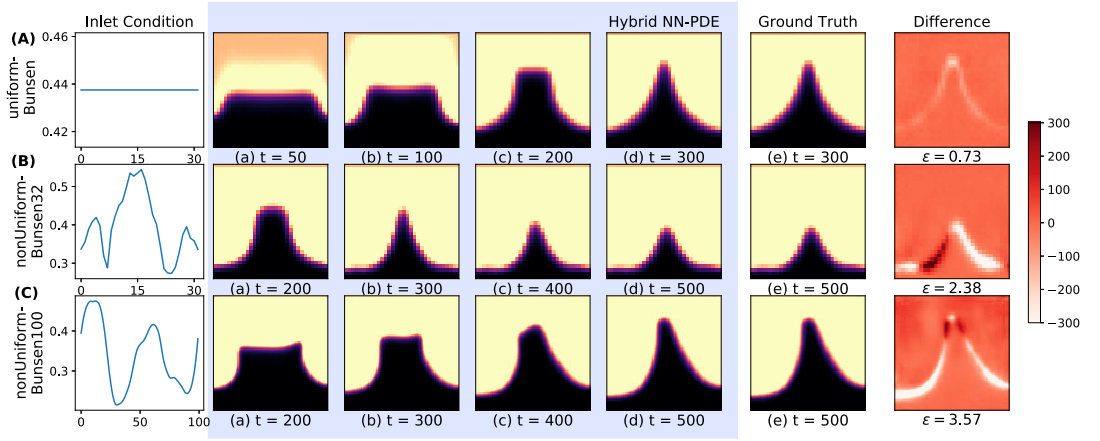

Figure 10. *Left: Inlet velocity profile used. (a–d) Temperature field prediction by hybrid NN-PDE approach over different steps given the inlet velocity profile. (e) Ground truth data. Right: Difference between ground truth data and hybrid NN-PDE output at last snapshot. Top to bottom: 32 × 32 resolution cases of (A) uniform-Bunsen, (B) nonUniform-Bunsen32, and (C) 100 × 100 testcase nonUniform-Bunsen100. ε represents the instantaneous MAPE.*

incomplete PDE solver. It accurately reproduces the complete PDE behavior. Figure 10B highlights this with a visualization of the hybrid NN-PDE model predictions for nonUniform-Bunsen32 case. The trained model accurately predicts the flame simulation over long roll-outs of 500 steps and achieves the complex flame shape with a low, instantaneous MAPE of 2.38%.

Finally, we showcase the ability of the hybrid approach to predict the temporal evolution of highly resolved flames with the nonUniform-Bunsen100 scenario. Despite the increased complexity of the larger resolution, it achieves a very good overall MAPE of 3.23% over 12 test cases of 500 simulation steps. Figure 10C shows an example of physically accurate predictions made by the hybrid NN-PDE model. We omit the evaluation of baselines for high-resolution cases as they do not succeed to model the flame dynamics for low-resolution cases.

In the following subsections, we present additional visualizations and a detailed comparison of the hybrid NN-PDE model against baseline approaches for different scenarios of Bunsen-type flames.

5.2.1. uniform-Bunsen

Figure 11 shows an example of the uniform-Bunsen case: a test case with $u_y = 0.375$ and $E = 0.95$. The constant inlet velocity results in a symmetric, Λ-shaped flame. Vertical, dotted, red line in Figure 11A helps to assess the symmetric nature of the flame. The PDD model predicts a thicker flame ($t = 50$) or a flame with a spurious tip ($t = 172$) or an asymmetric flame ($t = 225$). Snapshots of the hybrid NN-PDE model predictions in Figure 11A show that it adapts to this scenario very well and succeeds in obtaining the correct results for long-term forecasts of the temperature field. Furthermore, the flame shape and height are also better predicted, as shown in Figure 11A. Very low error levels in Figure 11B, such as $\epsilon = 1.26$ at $t = 300$, indicate that the hybrid model recovers the temperature field well.

5.2.2. nonUniform-Bunsen

Figure 12 compares the temporal predictions made by FNO, PDD, and the hybrid NN-PDE model with ground truth data for nonUniform-Bunsen32 case. Compared to Planar-v0 and uniform-Bunsen case, FNO predicts qualitatively better results for this case. Although still highly inaccurate, it predicts shapes that come closer to the target flame shapes for the later part of the simulation (for $t > 300$). This improvement might be due to the larger training dataset used for this scenario compared to the previous

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

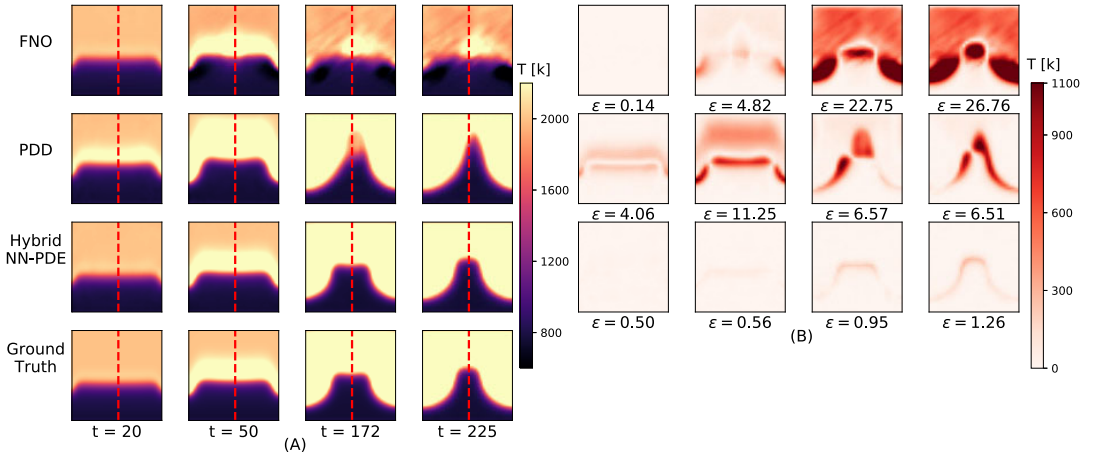

Figure 11. *uniform-Bunsen case with constant inlet velocity U = 0.375 and E = 0.95. (A) Temperature field predictions; (B) Absolute error between ground truth ($\mathcal{P}_c(T_t)$) and the output predicted*—from top to bottom top—by: FNO, purely data-driven approach and hybrid NN-PDE. Hybrid NN-PDE model predicts physically accurate evolution of the flame cases under study.

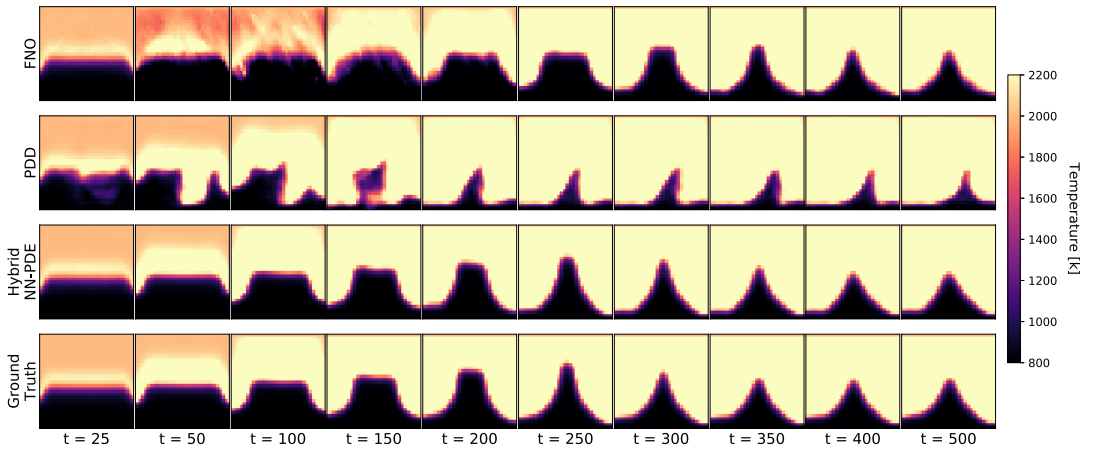

Figure 12. *nonUniform-Bunsen32: Comparison between different approaches—from top to bottom—FNO, PDD, hybrid NN-PDE for a test case. Hybrid approach predictions accurately match with the ground truth data over long-rollouts of 500 time steps.*

two scenarios. However, it fails to predict accurate temporal predictions over the entire simulation. The hybrid NN-PDE approach predicts the accurate flame evolution well. Additionally, we showcase the performance of the hybrid NN-PDE approach on two different test cases with complex flame shapes in Figure 13. The neural network model along with incomplete PDE solver reconstructs these complex flame shapes in an accurate manner.

For the nonUniform-Bunsen100 case, Figure 14 compares the predictions made by hybrid NN-PDE over {0,100,200,250,300,325,350,400,450,500} simulation steps with the ground truth data. It also shows the absolute difference between them. As the simulation progresses, higher errors are observed around the flame front. However, the hybrid NN-PDE approach captures the flame shape very accurately for longer roll-outs of 500 simulation steps. Currently, this approach uses training dataset similar to nonUniform-Bunsen32. Further improvements in accuracy can be achieved by increasing the training dataset size or training the hybrid model with longer look-ahead steps.

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

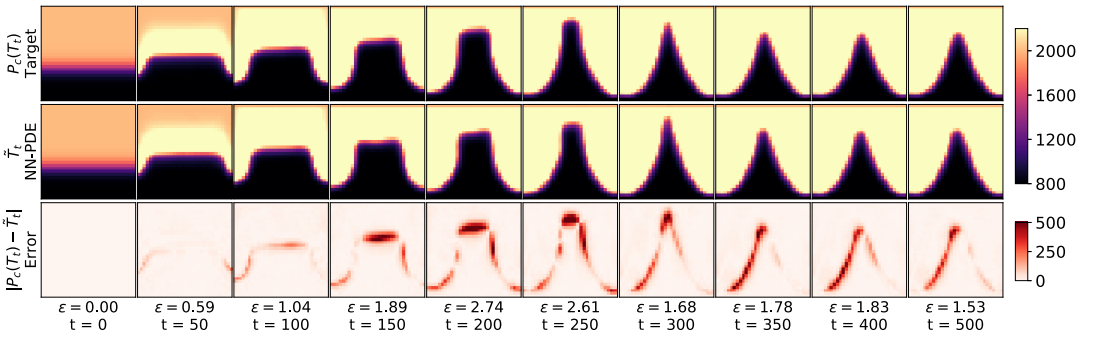

Figure 13. *nonUniform-Bunsen32: Visualization of reactive flow trajectories predicted by the hybrid approach for different test case.*

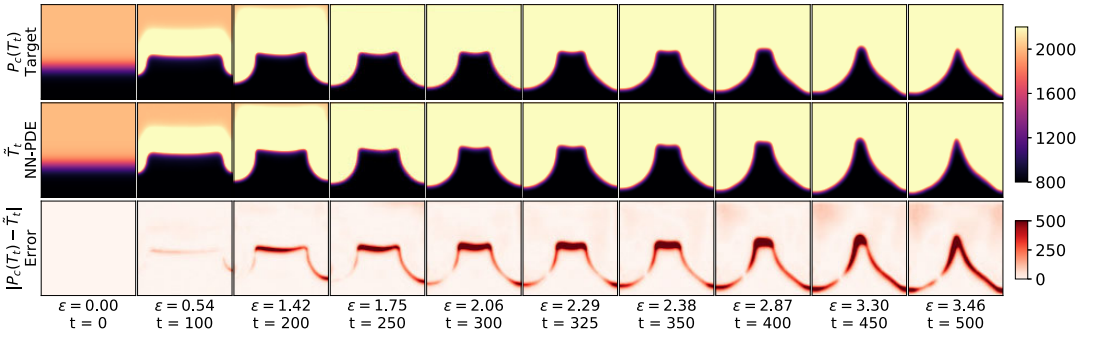

Figure 14. *nonUniform-Bunsen100: Comparison between ground truth data ($\mathcal{P}_c(T_t)$) and hybrid NN-PDE approach predictions ($\tilde{T}_t$) for a different test case of complex, high-resolution scenario.*

6. Discussion

In this section, we study the effect of temporal coarsening, incorrect PDE parameters, and longer look-ahead steps on predictions made by the hybrid NN-PDE solver.

6.1. Relaxing temporal stiffness in the PDE solver

Traditionally, the source terms and reaction rate terms involved in modeling the fast chemistry of reacting flows require the use of very small time-steps in simulations due to the stiffness of the chemical mechanisms. The incomplete PDE solver used in the hybrid NN-PDE approach, does not contain the source terms and reaction rate terms. Therefore, the time scales associated with the chemical reactions play a less important role in maintaining numerical stability, and it becomes possible for the solver to employ larger time steps. To illustrate this advantage, we train the neural network model with a time-step that is twice as large as the largest time-step Δt_c required for the complete PDE solver to yield a numerically stable result.

We mimic the setup described in Section 4, but now the incomplete PDE solver uses a time-step of $2\Delta t_c$, which is too large for the complete PDE solver by itself to converge to a solution. Figure 15 shows the results obtained from the models trained with a larger time-step are in good agreement with the target data for four different scenarios considered. The flame dynamics are predicted accurately while using less simulation steps. For the uniform-Bunsen case, the hybrid approach takes 8.23 ± 0.011s to infer one simulation run, whereas the complete PDE solver requires 15.21 ± 0.003s. Similar performance gains are observed across the other cases studied. Note that for the previous cases with a timestep size of one Δt_c, the trained model incurs a negligible runtime overhead. Even though incomplete PDE solvers are cheaper to

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

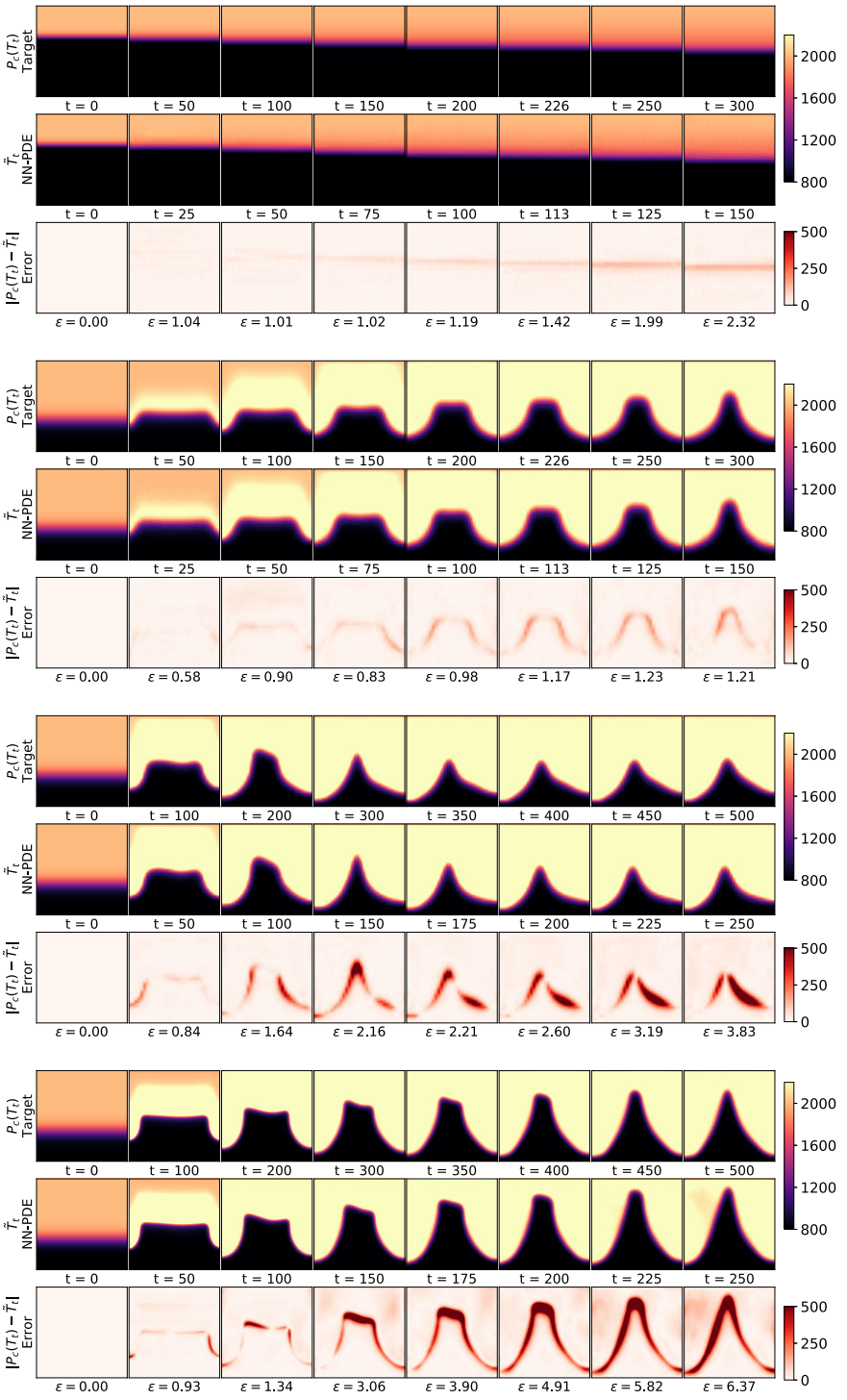

Figure 15. *Predictions made by hybrid NN-PDE model ($\tilde{T}_t$) with an incomplete PDE solver at twice the time-step of $2\Delta t_c$, are compared with the ground truth solutions coming from the complete PDE solver at time-step of Δt_c. We showcase the effectiveness of hybrid approach in relaxing temporal stiffness of the complete PDE solver on reactive flow cases of—from top to bottom—Planar-v0, uniform-Bunsen, nonUniform-Bunsen32 and nonUniform-Bunsen100 for different test cases.*

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

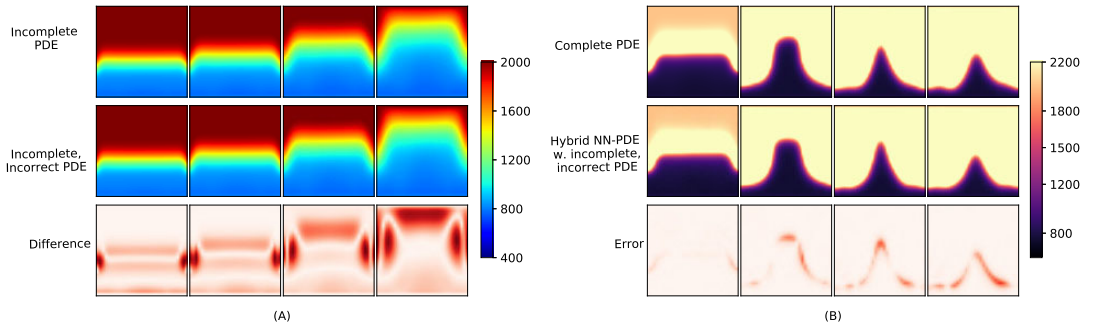

Figure 16. *Generalization to incorrect PDE parameters (A) visualization of differences in temperature fields due to incorrect parameters in incomplete PDE description. (B) Modified hybrid NN-PDE model, which combines the incomplete, incorrect PDE solver with neural network model is able to recover solutions of complete, correct PDE.*

run, as mentioned in Section 1, increasing the spatial or temporal resolution of the incomplete PDE solver would not converge to the solutions of the complete description.

The last column of Table 2 (*Hybrid NN-PDE-dt*) summarizes the errors of the large time-step approach for all four reactive cases considered. Despite an effectively doubled computational performance, this model achieves similar errors to the hybrid NN-PDE approach. This highlights the capabilities of learned, hybrid PDE solvers, which can produce these solutions without the stability problems exhibited by the complete PDE solver, while at the same time being more accurate than pure data-driven predictions.

6.2. Robustness of the hybrid solver

6.2.1. Generalization to incorrect PDE parameters

In real-world scenarios, even if the incomplete PDE system is observable, one may have only an approximate understanding of the physics behind it. This may lead to incorrect estimation of the parameters in the incomplete PDE solver. The proposed hybrid NN-PDE model is capable of completing the PDE description even if the underlying incomplete PDE solver has incorrect parameters. We refer to the incomplete solver with incorrect parameters as "incomplete, incorrect PDE." We experiment with a modified hybrid NN-PDE approach wherein we combine an incomplete, incorrect PDE solver with a neural network model. We assume that the known values of the incomplete PDE parameters in equation (5), strain rate tensor (τ), the diffusion coefficient of species k (D_k), and mixture thermal conductivity (λ), are incorrect. Figure 16A shows the difference between the temperature field evolution of the incomplete PDE solver with correct parameters ($[\tau, D_k, \lambda] = [0.1, 0.1, 0.1]$) and incorrect parameters ($[\tau, D_k, \lambda] = [0.05, 0.05, 0.05]$) at different time-steps. The hybrid NN-PDE combines this incomplete, incorrect PDE solver with the neural network model to obtain the solutions of the complete PDE solver with correct parameters. Figure 16B compares the flame dynamics predicted by this hybrid NN-PDE model with that of the complete, correct PDE solver. A good match is observed over various test cases. The hybrid model with incomplete, incorrect solver achieves an overall MAPE of 2.48 $\pm$ 1.20%, compared to the MAPE of 2.04 $\pm$ 1.39% for hybrid model with incomplete, correct PDE.

6.2.2. Effect of longer look-ahead steps

We evaluate the effect of varying look-ahead steps on the performance of the hybrid model. We show the comparison of models trained with $m = \{2, 4, 8, 16, 32\}$ for different cases in Figure 17. Models with smaller $m(2)$ do not learn to accurately correct the fields over long time and quickly diverge from the target simulation. Using larger m improves the quality of prediction drastically as the model learns the correction via the gradients over longer simulation steps. For Planar-v0 and uniform-Bunsen cases, iterating the NN and PDE solver for 32 time-steps improves the accuracy by 81.7% and 94%, respectively

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

compared to the $m=2$ model. For nonUniform-Bunsen32 and nonUniform-Bunsen100, the model performance improves by 84.2% and 70.9%, respectively, by using $m=32$ instead of $m=2$ model.

For some of error curves in Figure 17, significant error fluctuations are observed for lower simulation steps followed by recovery as the simulation progresses (e.g., Figure 17C, $m=4$). The evolution of the flame dynamics over time is highly nonlinear owing to the strong nonlinearity of the system where the premixed fuel and air mixture reacts to form the products. During the initial steps of this transient behavior, the distribution of flame temperature and flame interface varies rapidly. This leads to steep temperature gradients across the computational domain. As the simulation approaches toward the steady-state flame shape, the temperatures across the domain are in a similar range across different datasets. Therefore, small errors in flame temperature predictions and flame velocity predictions may lead to larger errors during the transient phase of the flame before it reaches a steady state.

6.2.3. Effect of training dataset size

We study the effect of training dataset size on the accuracy of PDD predictions. We train PDD models with different numbers of simulations in the training dataset. Each simulation contains 500 time-steps. Figure 18A shows the MAPE of PDD models trained with {4,8,12,16,24,32} training datasets, over a fixed testset. We compare the performance of these PDD models with an equivalent (trained with same look-ahead steps $m=2$) hybrid NN-PDE model, trained with 12 simulation sets. Figure 18A shows increasing the number of training sets has little or no effect on the prediction capabilities of the PDD models for nonUniform-Bunsen32 case. The hybrid model with 12 training sets achieves a MAPE of 12.49 ± 4.17%, an improvement of 38% over the PDD model with 32 training sets. The PDD models cannot achieve the same level of accuracy even in the presence of large amounts of data. This result strengthens the hypothesis that integrating the incomplete PDE solver into the neural network training yields a learning signal that fundamentally differs from that produced by training with precomputed data. The PDD models cannot achieve the same level of accuracy even in the presence of large amounts of data.

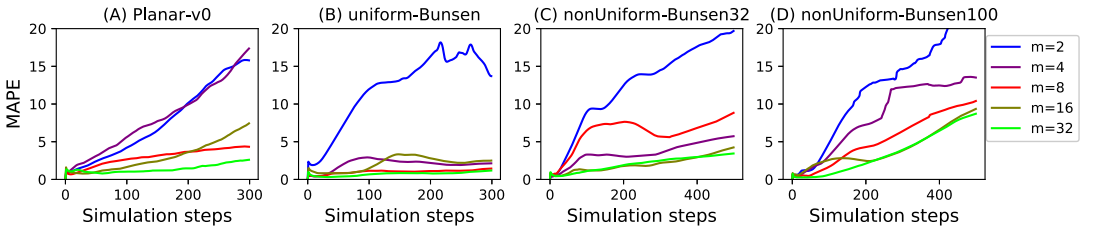

Figure 17. *Effect of longer look-ahead steps. MAPE of temperature field predictions by hybrid NN-PDE model, over all testsets. Models trained with higher look-ahead steps m accurately predict the temporal evolution of dynamics for longer duration across all cases considered.*

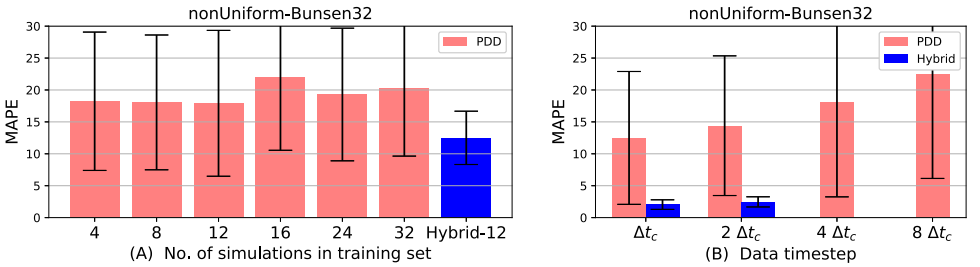

Figure 18. *(A) The effect of increasing training dataset size for the PDD approach, over fixed testsets, is compared with equivalent (trained with same look-ahead steps $m=2$) hybrid NN-PDE model. (B) Effect of temporal coarsening on PDD models trained with $m=32$ look-ahead steps.*

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

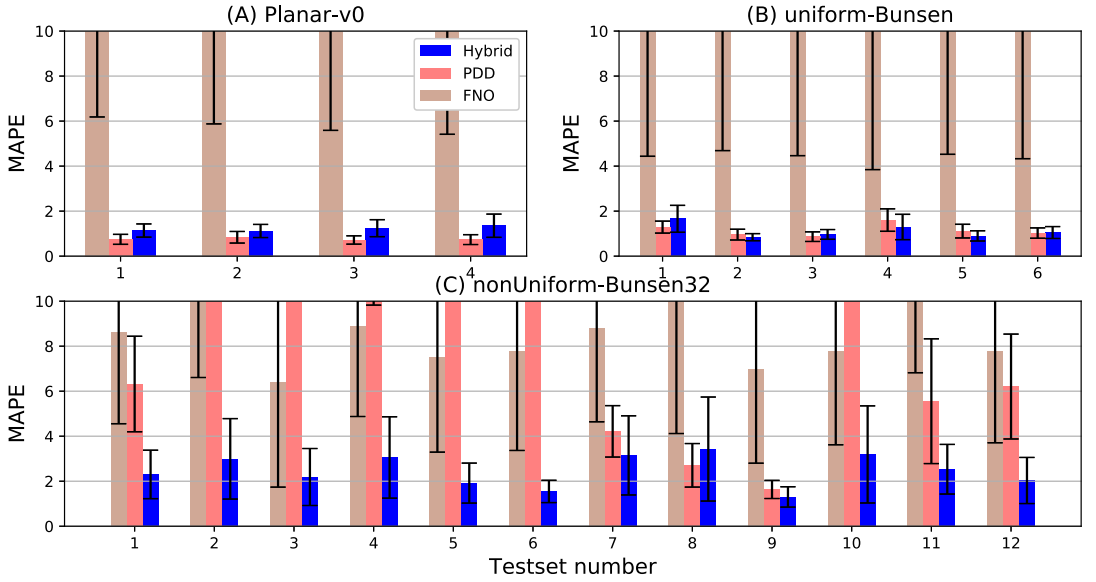

Figure 19. *Bar plot of MAPE of temperature field predictions by FNO, PDD, and hybrid NN-PDE model at time-step $2\Delta t_c$, for different testcases.*

6.2.4. Effect of temporal coarsening

Table 2 compares the MAPE of temperature field predictions at time-step Δt_c, for all three scenarios. The hybrid NN-PDE approach consistently performs better than both the baselines for all three scenarios considered. Higher standard deviations in Table 2 indicate the large differences in the prediction of flame temperatures for baseline approaches. The multi-physics systems we study provide a substantially more difficult environment than regular fluids: the chemical reactions are numerically very stiff, and the resulting dynamic interactions are difficult to capture by numerical solvers. To further illustrate the temporal behavior of the simulations, we perform additional experiments with the baseline approaches using twice the time-step size of the complete solver ($2\Delta t_c$). As shown in Figure 19, for the Planar-v0 and uniform-Bunsen case, PDD catches up with the hybrid approach whereas the PDD approach completely fails to predict the dynamics of the nonUniform-Bunsen32 case. We further investigate the performance of the PDD approach on the nonUniform-Bunsen32 case with larger time-steps (2 times, 4 times, and 8 times) as shown in Figure 18B. PDD achieves higher MAPE of 22.44% for $8\Delta t_c$ setup as compared to the MAPE of 12.48% for Δt_c setup. The PDD approach does not predict the dynamics accurately for any of the larger time-steps considered. We cannot showcase the performance of the hybrid NN-PDE solver for time steps greater than $2\Delta t_c$ as it is restricted by the underlying incomplete PDE solver. Figure 20 visualizes these results over different time-steps for one of the test cases. The predictions made by FNO, PDD, and hybrid NN-PDE model for nonUniform-Bunsen32 case at twice the time-step of $2\Delta t_c$, are compared with the ground truth solutions coming from the complete PDE solver at time-step of Δt_c. FNO and PDD (upper two rows) fail to recover the correct flame shapes over 250 simulation steps. The hybrid NN-PDE model (second row from below) predicts the flame shape accurately, thus relaxing the temporal stiffness of the complete PDE solver.

7. Flame shape control

To demonstrate the flexibility of the hybrid NN-PDE solver of the previous sections we consider a multi-physics system control problem. We test the joint applicability of a neural network with a pre-trained hybrid reactive flow solver to obtain a desired flame shape. We integrate the hybrid solver as a layer into the neural network model to enable the training of a flame shape controller. As a result, physics knowledge

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

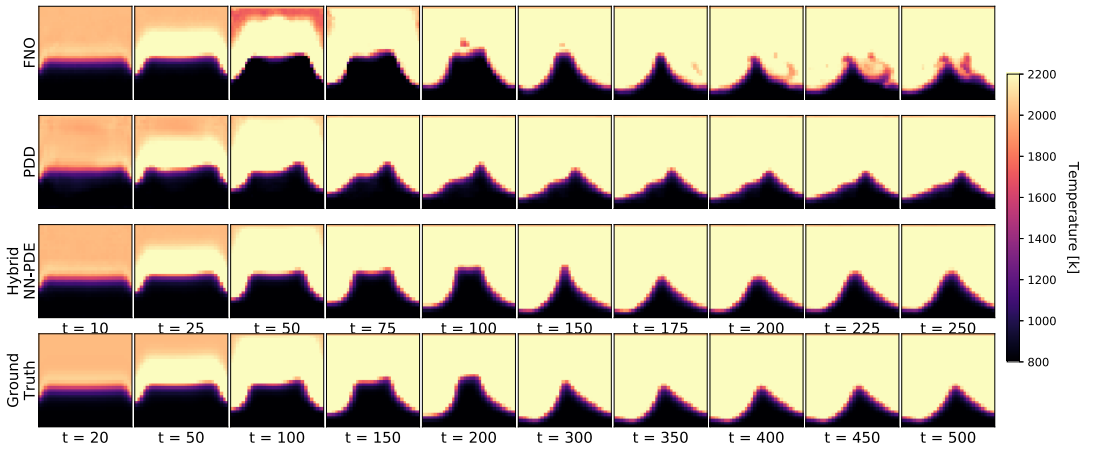

Figure 20. *Comparison between different approaches at time-step $2\Delta t_c$—from top to bottom—FNO, PDD, hybrid NN-PDE for nonUniform-Bunsen32 test case. Hybrid approach predictions accurately match with the ground truth data over long roll-outs.*

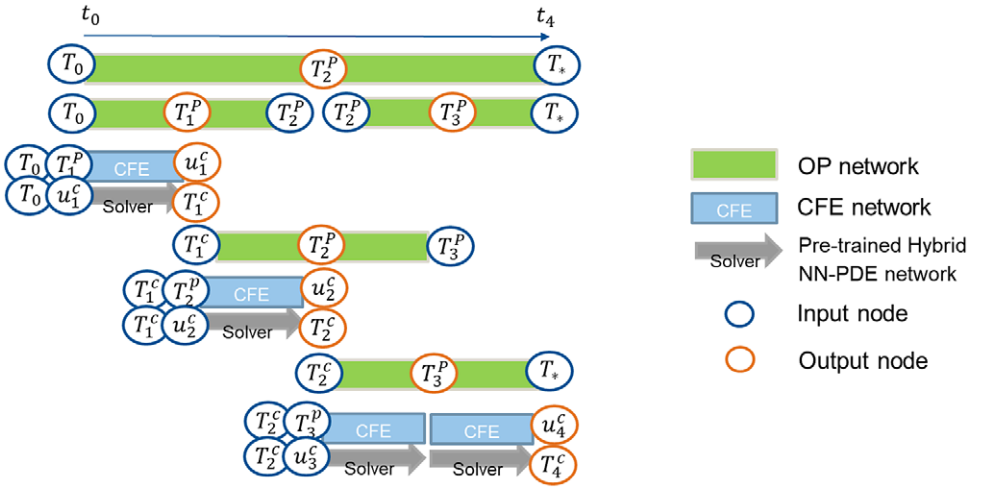

Figure 21. *Details of the OP-CFE prediction sequence.*

from the hybrid solver can be embedded into the optimization problem, by matching observations coming from the solver. The hybrid solver provides the agent with feedback of how interactions at any point in time affect the flame shape. We consider a physical system of reactive flows where the agent can interact with the system by controlling the inlet velocity of fuel-air flow.

We use a predictor–corrector approach (Holl et al., 2020) with a supervised loss function and control sequence refinement for the nonUniform-Bunsen case with 32 × 32 resolution. A pre-trained hybrid solver from Section 5.2.2 is used as a reactive flow solver. This hybrid model consists of incomplete PDE solver along with an UNet model detailed in Sections 3.3 and 3.4 and Figures 2 and 3. It is first trained using training instances of the uniform-Bunsen32 case described in Section 4. Additionally, it is further trained on the training data of flame shape transitions from an arbitrary flame shape to any other arbitrary flame shape. Once trained, the weights of the hybrid solver are frozen and used as a layer into the predictor–corrector approach. It is differentiable by construction, when combined with the deep neural networks is shown to learn a policy to control flame shapes. The learning objective is to arrive at a target

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

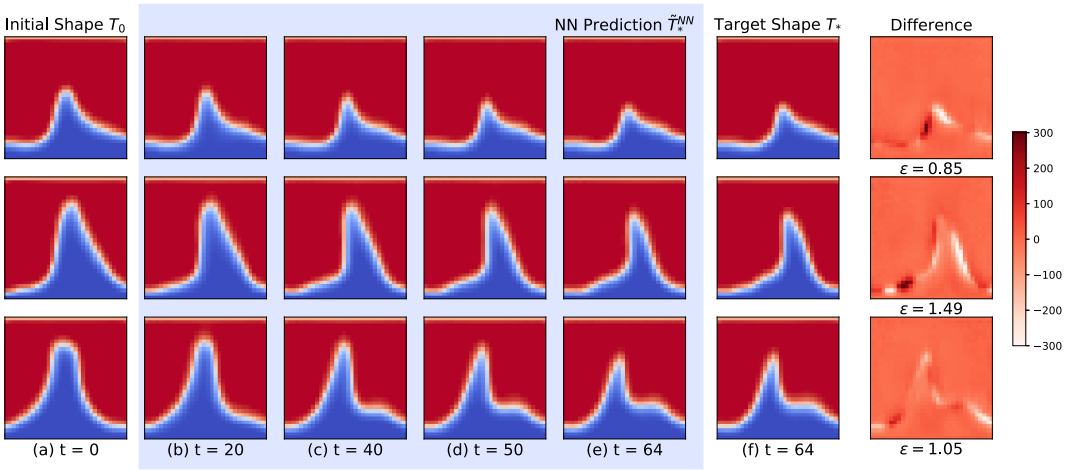

Figure 22. *Temperature field predictions are achieved by the neural network model $\tilde{T}_*^{\mathrm{NN}}$ from given initial flame shape T_0 to achieve target flame shape $\tilde{T}_*$ by controlling flow velocity.*

temperature field (T_*) with the desired flame shape from an arbitrary, given initial configuration (T_0) by controlling the velocity field (u_i^c). We consider a real-world setting where only few states of the reactive flow system are observable and controllable: the temperature and mass-fraction fields are observable, and only the inlet of the velocity field is controllable. The predictor–corrector approach consists of two neural network models. First, an observation predictor network (OP), predicts the intermediate states of the observable quantity, that is, temperature in this case (T_i^p). The corrector (CFE) network predicts the control force (u_i^c) required to follow the trajectory predicted by OP network. The inlet velocity field predicted by the CFE network is used by the pre-trained hybrid solver to compute the corrected temperature field (T_i^c). Figure 21 shows the sequence of OP-CFE network used to control the flame shape T_* given the initial flame shape T_0, over the trajectory of four time-steps. The OP network is modeled as a temporal hierarchical process to incorporate the knowledge about longer time spans. Therefore, OP networks are trained to predict the optimal center point between two time-steps instead of predicting the state of the next time step. The recursive call to the OP network enables the prediction of observable quantities at every time step (T_i^p). Using T_{i-1} and OP prediction at step i, that is, T_i^p, CFE network predicts the control force at step i, u_i^c. Once u_i^c is predicted, learned hybrid solver can be used to advance the simulation to next time step and obtain the actual state T_i^c. Using this sequence of OP-CFE network, we can obtain T_n^c with n CFE evaluations, which matches target flame shape T_* very closely. This sequence can be generalized to any arbitrarily long length of sequences. OP-CFE networks are trained using the supervised loss function on the target temperature field achieved and inlet velocity predicted, respectively. The loss function formulations for OP and CFE networks are,

$$L_{\mathrm{op}} = \left| \mathrm{OP}\left[T_i, T_j\right] - T^{GT}\left(\frac{i+j}{j}\right) \right|^2 \qquad L_{\mathrm{CFE}} = \sum_{i=1}^{n} \left| u_i^c - u_i^* \right|^2, \tag{9}$$

where T^{GT} and u^* indicate the ground truth temperature field and inlet velocity, respectively. The simplest technique to obtain the optimal trajectory, given an initial state is to use the chain of CFE networks followed by the solver. CFE, modeled using deep neural network, predicts the control force required to predict the transition to next observable state. Therefore, as a baseline model, we consider the CFE only approach. All neural networks used in this work are modified UNet architectures (Ronneberger et al., 2015).

Figure 22b–E shows examples of the transition achieved and compares the target flame shape obtained using the neural network model with the target flame shape (T_*). All three cases are test cases, that is, were not seen at training time, and have a sequence length of 64 simulation steps. It is visible that the neural

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

network model succeeds at controlling the inlet velocity of fuel-air mixture flow to obtain the desired flame shapes. To quantify these results, the learned model achieves MAPE of 1.34 ± 0.41% over 50 test cases considered. It achieves an improvement of 65.7% compared to a baseline model of the CFE only approach, without a predictor network for the long-term prediction of temperature fields. This baseline model has a MAPE of 3.91 ± 2.19%. This case highlights the capabilities of differentiable hybrid NN-PDE multi-physics solvers. Specifically, the hybrid solver helps the OP-CFE networks to yield controllers that steer the complex physics of the reactive flow over long time spans.

8. Conclusion

In this work, we proposed a hybrid NN-PDE solver which learns to complete an incomplete PDE solver in the context of a multi-physics system, namely the case of a reacting flow. The trained neural network model works alongside an incomplete solver to accurately account for and predict the effects of unknown physics. We compare its performance with two baselines—a PDD approach and state-of-the-art FNO approach. We test it on the cases of reacting flows of increasing complexity. The qualitative and quantitative performance of the presented hybrid NN-PDE approach is found to be superior compared to these baselines to predict the long-term temporal evolution of the reactive flows. The incomplete PDE description helps the neural network model recover the target simulation with a significantly improved accuracy. This hybrid NN-PDE model is able to predict the correct evolution of the important features of the multi-physics system (in our case, the flame interface and flame shape) for longer simulation steps in all scenarios and is able to generalize to other (initial and boundary) conditions of the system. This is demonstrated by variations in the equivalence ratio and inlet velocity forcing. We also show that such hybrid approach may have the added benefits of allowing to relax numerical constraints linked to the potential stiffness of the unknown physics. The learned hybrid NN-PDE solver is successfully adapted to enable flame shape control by combining with deep neural networks.

Our work represents a stepping stone for numerous avenues of future work. While we demonstrated the applicability of the proposed approach to the complex case of a reacting flow (with strong nonlinear interactions between chemistry and velocity), the proposed hybrid NN-PDE solver could be further utilized to predict and control the dynamics of other tightly coupled multi-physics systems (Levy, 1999; Dowell and Hall, 2001). In addition, applying the proposed approach to other cases, where a limited amount of real data is available would be a very interesting step. Especially if the data is very sparse and contains noise, learning meaningful corrections could become excessively challenging. Performing such an assessment will be explored in future work. We note that the proposed approach does not seek to replace the classical physical simulation approaches. However, it is an important step in the direction of harnessing the capabilities of neural network models using the knowledge of PDEs for complex multi-physical systems.

Author contribution. Conceptualization: N.T.; Data curation: N.N.T.; Formal analysis: N.N.T.; Investigation: N.N.T.; Methodology: N.N.T.; Software: N.N.T.; Supervision: N.T., N.A.K.D., C.F.S.; Validation: N.N.T.; Visualization: N.N.T.; Writing—original draft: N.N.T.; Writing—review and editing: N.N.T., N.A.K.D., C.F.S., N.T. All authors approved the final submitted draft.

Competing interest. The authors declare no competing interests.

Data availability statement. The code to generate the data and train models that support the findings of this study are openly available at https://github.com/tum-pbs/Hybrid-Solver-for-Reactive-Flows.

Funding statement. N.N.T acknowledges the financial support of the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No. 766264 and the ERC Consolidator Grant *SpaTe* (CoG-2019-863850).

Ethical standard. The research meets all ethical guidelines, including adherence to the legal requirements of the study country.

Supplementary material. The supplementary material for this article can be found at https://doi.org/10.1017/dce.2023.20.

References

Ajuria Illarramendi E, **Alguacil A**, **Bauerheim M**, **Misdariis A**, **Cuenot B and Benazera E** (2020) Towards an hybrid computational strategy based on deep learning for incompressible flows. In *AIAA AVIATION 2020 FORUM*. Reston VA: American Institute of Aeronautics and Astronautics, p. 3058.

Bar-Sinai Y, **Hoyer S**, **Hickey J and Brenner MP** (2019) Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences 116*(31), 15344–15349.

Belbute-Peres FA, **Economon T and Kolter Z** (2020) Combining differentiable PDE solvers and graph neural networks for fluid flow prediction. *International Conference on Machine Learning 119*, 2402–2411.

Benra F-K, **Dohmen HJ**, **Pei J**, **Schuster S and Wan B** (2011) A comparison of one-way and two-way coupling methods for numerical analysis of fluid-structure interactions. *Journal of Applied Mathematics 2011*, 853560.

Bhatnagar S, **Afshar Y**, **Pan S**, **Duraisamy K and Kaushik S** (2019) Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics 64*, 525–545.

Bhattacharya K, **Hosseini B**, **Kovachki NB and Stuart AM** (2021) Model reduction and neural networks for parametric pdes. *The SMAI Journal of Computational mathematics 7*, 121–157.

Chen CJ (1997) *Fundamentals of Turbulence Modelling*. Boca Raton, FL: CRC Press.

Chen RT, **Rubanova Y**, **Bettencourt J and Duvenaud D** (2018) Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Red Hook, NY: ACM, pp. 6572–6583.

Dowell EH and Hall KC (2001) Modeling of fluid-structure interaction. *Annual Review of Fluid Mechanics 33*, 445.

Dresdner G, **Kochkov D**, **Norgaard P**, **Zepeda-Núñez L**, **Smith JA**, **Brenner MP and Hoyer S** (2022) Learning to correct spectral methods for simulating turbulent flows. *arXiv preprint arXiv:2207.00556*.

Dwivedi V and Srinivasan B (2020) Physics informed extreme learning machine (pielm) – A rapid method for the numerical solution of partial differential equations. *Neurocomputing 391*, 96–118.

Economon TD, **Palacios F**, **Copeland SR**, **Lukaczyk TW and Alonso JJ** (2016) Su2: An open-source suite for multiphysics simulation and design. *AIAA Journal 54*(3), 828–846.

Fukami K, **Fukagata K and Taira K** (2019) Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics 870*, 106–120.

Fuks O and Tchelepi HA (2020) Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing 1*(1), 1–15.

Gamezo VN, **Khokhlov AM**, **Oran ES**, **Chtchelkanova AY and Rosenberg RO** (2003) Thermonuclear supernovae: Simulations of the deflagration stage and their implications. *Science 299*(5603), 77–81.

Greydanus S, **Dzamba M and Yosinski J** (2019) Hamiltonian neural networks. *Advances in Neural Information Processing Systems 32*, 15379–15389.

Gruber A, **Richardson ES**, **Aditya K and Chen JH** (2018) Direct numerical simulations of premixed and stratified flame propagation in turbulent channel flow. *Physical Review Fluids 3*(11), 110507.

Guo X, **Li W and Iorio F** (2016) Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: ACM, pp. 481–490.

Han J, **Jentzen A and Weinan E** (2018) Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences 115*(34), 8505–8510.

Harlow FH and Welch JE (1965) Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The Physics of Fluids 8*(12), 2182–2189.

He K, **Zhang X**, **Ren S and Sun J** (2016) Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Las Vegas, NV: IEEE, pp. 770–778.

Holl P, **Koltun V**, **Um K and Thuerey N** (2020) Phiflow: A differentiable PDE solving framework for deep learning via physical simulations. *NeurIPS Workshop*.

Holl P, **Thuerey N and Koltun V** (2020) *Learning to control PDEs with differentiable physics. In 8th International Conference on Learning Representations, April 26–30, 2020*. Addis Ababa, Ethiopia: ICLR. Available at https://openreview.net/forum?id=HyeSin4FPB. (accessed 9 September 2022)

Illarramendi EA, **Bauerheim M and Cuenot B** (2022) Performance and accuracy assessments of an incompressible fluid solver coupled with a deep convolutional neural network. *Data-Centric Engineering 3*, e2.

Jacobson MZ (1999) *Fundamentals of Atmospheric Modeling*. Cambridge: Cambridge University Press.

Jaensch S, **Merk M**, **Gopalakrishnan E**, **Bomberg S**, **Emmert T**, **Sujith R and Polifke W** (2017) Hybrid CFD/low-order modeling of nonlinear thermoacoustic oscillations. *Proceedings of the Combustion Institute 36*(3), 3827–3834.

Kalnay E (2003) *Atmospheric Modeling, Data Assimilation and Predictability*. Cambridge: Cambridge University Press.

Kim B, **Azevedo VC**, **Thuerey N**, **Kim T**, **Gross M and Solenthaler B** (2019) Deep fluids: A generative network for parameterized fluid simulations. *Computer Graphics Forum 38*, 59–70.

Kingma DP and Ba J (2015) Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Knio OM, **Najm HN and Wyckoff PS** (1999) A semi-implicit numerical scheme for reacting flow: II. Stiff, operator-split formulation. *Journal of Computational Physics 154*(2), 428–467.

Kochkov D, **Smith JA**, **Alieva A**, **Wang Q**, **Brenner MP and Hoyer S** (2021) Machine learning accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences 118*(21), e2101784118.

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

Lagaris IE, **Likas A and Fotiadis DI** (1998) Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks 9*(5), 987–1000.

Lapeyre CJ, **Misdariis A**, **Cazard N**, **Veynante D and Poinsot T** (2019) Training convolutional neural networks to estimate turbulent sub-grid scale reaction rates. *Combustion and Flame 203*, 255–264.

Levy S (1999) *Two-Phase Flow in Complex Systems*. Hoboken, NJ: John Wiley & Sons.

Li Z, **Kovachki N**, **Azizzadenesheli K**, **Liu B**, **Bhattacharya K**, **Stuart A and Anandkumar A** (2020a) *Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895*.

Li Z, **Kovachki N**, **Azizzadenesheli K**, **Liu B**, **Bhattacharya K**, **Stuart A and Anandkumar A** (2020b) Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*.

Li Z, **Zheng H**, **Kovachki N**, **Jin D**, **Chen H**, **Liu B**, **Azizzadenesheli K and Anandkumar A** (2021) Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*.

Lieuwen TC (2012) *Unsteady Combustor Physics*. Cambridge: Cambridge University Press.

Lieuwen TC and Yang V (2005) *Combustion Instabilities in Gas Turbine Engines: Operational Experience, Fundamental Mechanisms, and Modeling*. Reston, VA: American Institute of Aeronautics and Astronautics.

List B, **Chen L-W and Thuerey N** (2022) Learned turbulence modelling with differentiable fluid solvers. *arXiv preprint arXiv:2202.06988*.

Liu Z and Liu Z (2018) *Multiphysics in Porous Materials*. Cham: Springer.

Long Z, **Lu Y**, **Ma X and Dong B** (2018) PDE-net: Learning PDEs from data. *International Conference on Machine Learning 80*, 3208–3216.

Lu L, **Jin P**, **Pang G**, **Zhang Z and Karniadakis GE** (2021) Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence 3*(3), 218–229.

Mitani T (1980) Propagation velocities of two-reactant flames. *Combustion Science and Technology 21*(3–4), 175–177.

Najm HN, **Wyckoff PS and Knio OM** (1998) A semi-implicit numerical scheme for reacting flow: I. Stiff chemistry. *Journal of Computational Physics 143*(2), 381–402.

Özbay AG, **Hamzehloo A**, **Laizet S**, **Tzirakis P**, **Rizos G and Schuller B** (2021) Poisson cnn: Convolutional neural networks for the solution of the poisson equation on a cartesian mesh. *Data-Centric Engineering 2*, e6. https://doi.org/10.1017/dce.2021.7.

Patel RG, **Trask NA**, **Wood MA and Cyr EC** (2021) A physics-informed operator regression framework for extracting data-driven continuum models. *Computer Methods in Applied Mechanics and Engineering 373*, 113500.

Pathak J, **Mustafa M**, **Kashinath K**, **Motheau E**, **Kurth T and Day M** (2020) Using machine learning to augment coarse-grid computational fluid dynamics simulations. *arXiv preprint arXiv:2010.00072*.

Poinsot T and Veynante D (2005) *Theoretical and Numerical Combustion*. Philadelphia, PA: RT Edwards, Inc.

Pontryagin LS (1987) *Mathematical Theory of Optimal Processes*. Boca Raton, FL: CRC Press.

Pope SB (2000) *Turbulent Flows*. Cambridge: Cambridge University Press.

Rackauckas C, **Ma Y**, **Martensen J**, **Warner C**, **Zubov K**, **Supekar R**, **Skinner D**, **Ramadhan A and Edelman A** (2020) Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*.

Raissi M, **Perdikaris P and Karniadakis GE** (2019) 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.

Rhie CM and Chow W-L (1983) Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA Journal 21*(11), 1525–1532.

Rolnick D, **Donti PL**, **Kaack LH**, **Kochanski K**, **Lacoste A**, **Sankaran K**, **Ross AS**, **Milojevic-Dupont N**, **Jaques N**, **Waldman-Brown A**, **Luccioni AS**, **Maharaj T**, **Sherwin ED**, **Karthik Mukkavilli S**, **Kording KP**, **Gomes CP**, **Ng AY**, **Hassabis D**, **Platt JC**, **Creutzig F**, **Chayes J and Bengio Y** (2022) Tackling climate change with machine learning. *ACM Computing Surveys (CSUR) 55*(2), 1–96.

Ronneberger O, **Fischer P and Brox T** (2015) U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Cham: Springer, pp. 234–241.

Sirignano J, **MacArt JF and Freund JB** (2020) Dpm:Adeep learning PDE augmentation method with application to large-eddy simulation. *Journal of Computational Physics 423*, 109811.

Stachenfeld K, **Fielding DB**, **Kochkov D**, **Cranmer M**, **Pfaff T**, **Godwin J**, **Cui C**, **Ho S**, **Battaglia P and Sanchez-Gonzalez A** (2022) Learned simulators for turbulence. *International Conference on Learning Representations*. Available at https://openreview.net/forum?id=msRBojTz-Nh. (accessed 15 January 2023)

Takeishi N and Kalousis A (2021) Physics-integrated variational autoencoders for robust and interpretable generative modeling. *Advances in Neural Information Processing Systems 34*, 14809–14821.

Thuerey N, **Holl P**, **Mueller M**, **Schnell P**, **Trost F and Um K** (2021) Physics-based deep learning. *arXiv preprint arXiv:2109.05237*.

Thuerey N, **Weißenow K**, **Prantl L and Hu X** (2020) Deep learning methods for Reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal 58*(1), 25–36.

Tompson J, **Schlachter K**, **Sprechmann P and Perlin K** (2017) Accelerating Eulerian fluid simulation with convolutional networks. In *International Conference on Machine Learning*. Sydney: PMLR, pp. 3424–3433.

Um K, **Brand R**, **Yun F**, **Holl P and Thuerey N** (2020) Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers. *Neural Information Processing Systems (NeurIPS) 33*, 6111–6122.

https://doi.org/10.1017/dce.2023.20 Published online by Cambridge University Press

Wang R, **Kashinath K**, **Mustafa M**, **Albert A and Yu R** (2020) Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York: ACM, pp. 1457–1466.

Wang S, **Wang H and Perdikaris P** (2021) Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science Advances 7*(40), eabi8605.

Wanner G and Hairer E (1996) *Solving ordinary differential equations II* (Vol. *375*). Berlin: Springer.

Werbos PJ (1990) Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE 78*(10), 1550–1560.

Wheeler AA, **Boettinger WJ and McFadden GB** (1992) Phase-field model for isothermal phase transitions in binary alloys. *Physical Review A 45*(10), 7424.

Williams FA (1985) *Combustion Theory*. California: Cummings.

Yadav V, **Casel M and Ghani A** (2022) Physics-informed recurrent neural networks for linear and nonlinear flame dynamics. *Proceedings of the Combustion Institute 39*, 1597–1606.

Yin Y, **Le Guen V**, **Dona J**, **de Bézenac E**, **Ayed I**, **Thome N and Gallinari P** (2021) Augmenting physical models with deep networks for complex dynamics forecasting. *Journal of Statistical Mechanics: Theory and Experiment 2021*(12), 124012.

Zafar MI, **Choudhari MM**, **Paredes P and Xiao H** (2021) Recurrent neural network for end-to-end modeling of laminar-turbulent transition. *Data-Centric Engineering 2*, e17.

Cite this article: Tathawadekar NN, Doan NAK, Silva CF and Thuerey N (2023). Incomplete to complete multiphysics forecasting: a hybrid approach for learning unknown phenomena. *Data-Centric Engineering*, 4, e27. doi:10.1017/dce.2023.20