

ABSTRACT

SACHDEVA, KUNAL. Accelerating Reactive Flow Simulations with Deep Operator Networks: Integration of DeepONets with PeleLMEx. (Under the direction of Tarek Echehki).

This thesis presents a data-driven framework for accelerating chemical kinetics integration in low Mach number reactive flow simulations, specifically targeting the computationally intensive chemistry modules in high-fidelity CFD solvers. The proposed methodology leverages the React-DeepONet architecture—a neural operator-based surrogate model capable of learning the temporal evolution of a reduced thermochemical state vector. This surrogate replaces the traditional stiff ODE solvers, such as CVODE, used within the PeleLMEx solver for chemical source term integration.

The framework decomposes the thermochemical state into representative and non-representative species. The representative scalars—including temperature and six key chemical species—are predicted using React-DeepONet, while the remaining species are reconstructed via a separate feedforward network called RecNet. This decomposition drastically reduces the dimensionality of the learning problem while preserving the fidelity of the complete chemical state.

To integrate this surrogate into the C++-based PeleLMEx environment, substantial engineering was performed to enable efficient inference using the Python-based JAX model. A hybrid C–Python interface using the NumPy C API was developed to allow zero-copy memory sharing and minimal overhead. Additionally, JAX's `jit` compilation was utilized to optimize model prediction speed, significantly reducing the runtime overhead of ML inference during simulation.

The framework was validated on a two-dimensional backward-facing step (BSF) flame configuration using a DRM-19 methane-air chemical mechanism. The simulation setup included detailed transient phenomena and flame-vortex interactions, which served as a rigorous test for the ML models. Offline validation of the React-DeepONet predictions against 0D Cantera-generated trajectories demonstrated excellent agreement in both the representative and recon-

structed species. In-situ deployment within PeleLMeX further confirmed the model's predictive power, with React-DeepONet capturing complex spatio-temporal structures in temperature and species fields with high accuracy.

Contour plots of key variables—temperature, CH_4 , CO , and HO_2 —were compared between the CVODE and DeepONet-integrated simulations. The results revealed minimal discrepancies, even during the formation of flame roll-up and merging events, indicating robust temporal stability. The HO_2 predictions, inferred via the RecNet, retained sharp post-flame gradients and interfaces, validating the reconstruction strategy. Despite using only a moderate-sized mechanism (DRM-19), the framework achieved an 11% speedup for the BSF problem. This performance gain, while modest, showcases the potential for much larger benefits in simulations involving complex fuels and larger reaction mechanisms.

Overall, the React-DeepONet framework demonstrates the feasibility of integrating operator learning-based surrogates into existing HPC-ready solvers. It marks a significant step toward scalable, generalizable, and efficient chemical kinetics modeling in turbulent reacting flows.

© Copyright 2025 by Kunal Sachdeva

All Rights Reserved

Accelerating Reactive Flow Simulations with Deep Operator Networks: Integration of
DeepONets with PeleLMEX

by
Kunal Sachdeva

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Aerospace Engineering

Raleigh, North Carolina
2025

APPROVED BY:

Chi-An Yeh

Srinath Ekkad

Tarek Echehki
Chair of Advisory Committee

DEDICATION

To my mother, father, and my dear sister:
whose unwavering love, sacrifices, and support have been the foundation of my journey. This
achievement is as much yours as it is mine.

BIOGRAPHY

Kunal was born in Chandigarh, located in the northern part of India. He completed his schooling at Tender Heart School, a nearby institution known for its academic environment. In 2023, he earned his Bachelor of Technology degree in Aerospace Engineering from Punjab Engineering College (Deemed to be University), in Chandigarh. Motivated by a passion for contributing to the field of science and technology, Kunal pursued graduate studies in the United States. He enrolled at North Carolina State University (NCSU) in Fall 2023 to pursue a Master of Science in Aerospace Engineering, immediately following the completion of his undergraduate degree. In Summer 2024, Kunal began his master's thesis research under the guidance of Dr. Tarek Echehki. His work focuses on accelerating reactive computational fluid dynamics (CFD) simulations using deep learning methods and integrating them into an open-source, high-fidelity CFD code. Kunal will graduate with a Master of Science degree in Aerospace Engineering in Spring 2025. He will begin his Ph.D. in Mechanical Engineering at Mississippi State University starting in Summer 2025.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Dr. Tarek Echekki, for his invaluable guidance, support, and encouragement throughout this research journey. When my former advisor left the university unexpectedly, I approached Dr. Echekki about one and a half years ago to continue my master's thesis under his supervision. Despite my lack of prior background in machine learning at that time, Dr. Echekki graciously accepted me as a student. I remain deeply grateful for his patience, mentorship, and belief in my potential, which played a critical role in shaping this work. I would also like to sincerely thank my committee members, Dr. Chi-An Yeh and Dr. Srinath Ekkad, for their valuable insights, feedback, and support during the course of this research. Additionally, I extend my gratitude to Dr. Veeraraghava Raju Hasti, my former advisor, for helping me kickstart my research journey and providing me with early experience in multiphase flow simulations.

I am deeply thankful to my dear friend, Utsavkumar Lal, a computer science graduate student at NCSU, for his unwavering support during one of the most challenging phases of my project. For several months, we met regularly, working through countless compilation issues until they were finally resolved by the end of February. I am profoundly grateful for Utsavkumar's patience, dedication, and technical guidance throughout this long process.

I would also like to thank Dr. Bruce Perry and Dr. Marc Day at the National Renewable Energy Laboratory (NREL) for their technical assistance and guidance. Their prompt and thoughtful responses to my questions, no matter how simple they might have seemed, greatly helped me in moving my work forward.

I would also like to thank my labmates and dear friends, Dakshaka and Sanket, for their help, useful advice, and for making my time in the lab enjoyable. A special thanks to my colleague Anuj for his help in my work and his constant support. Their support and encouragement made my research journey much easier and more positive.

I am grateful for my friends in Raleigh and the graduate students in the department —

Pranami, Sakshi, Mukul, Shridhar, Sarvesh, Sameer, Venkat, Aniketh, Kireeti, Varun, Pinaki, Het, Tae, Rohan, and Tirth — for making my master’s journey memorable. Their friendship and support made my time outside of research enjoyable and special. A special thanks to my flatmates, Venkat and Dhyey, for always keeping a chill and positive atmosphere at home, making everyday life much easier. I am also grateful to my undergraduate friends, Abhinav, Sahil, and Yuganshu, who were always just a call away and whose support meant a lot to me throughout this time.

A very special thank you goes to my girlfriend, Shrishty, for being my biggest source of strength throughout this journey. From helping me with coding doubts to lifting my spirits with encouragement, good food, and endless patience, she was always there. Her love and belief in me made my master’s journey truly joyful, and I am incredibly grateful for her support.

Finally, I would like to thank my parents for their sacrifices, constant encouragement, and prayers that gave me strength throughout this journey. I am also very grateful to my sister, Muskaan, for her endless support and motivation, even from afar. Their love has been the foundation behind everything I have achieved.

Above all, I thank God for giving me the strength, patience, and blessings to complete this journey.

TABLE OF CONTENTS

List of Tables	vii
List of Figures	viii
Chapter 1 Introduction	1
1.1 Motivation	1
1.2 Background and Challenges in Reaction Integration	3
1.3 Objective of This Work	5
1.4 Thesis Outline	5
Chapter 2 Methodology	7
2.1 Overview of PeleLMeX and Its Reaction Integration Approach	7
2.1.1 PeleLMeX Submodules	8
2.1.2 Governing Equations for Low Mach Number Flow	10
2.1.3 Reaction Integration Approach in PeleLMeX	11
2.2 Deep Operator Networks (DeepONets)	13
2.2.1 Methodology	14
2.2.2 Reduction of the solution vector of thermochemical state using representative scalars	17
2.2.3 Reconstruction of the remaining thermochemical scalars	18
2.3 Training Data Generation and Model Training Mechanism	19
2.4 DeepONet integration in PeleLMeX	20
2.4.1 Integration Challenges and Technical Constraints	20
2.4.2 Implementation Workflow and Runtime Integration	22
Chapter 3 Results and Validation	27
3.1 Background	27
3.2 Case Setup, Data Generation, and Model Training Summary	29
3.2.1 Simulation Configuration	29
3.2.2 Data Sampling and 0D Evolutions	31
3.2.3 Model Training and Convergence Behavior	34
3.2.4 Offline Predictions	36
3.3 Results	39
3.3.1 Scalar Field Comparisons	40
3.3.2 Computational Performance and Speed-Up	53
3.4 Conclusion	54
Chapter 4 Conclusions and Future Works	56
4.1 Summary	56
4.2 Future Work	58
References	59

LIST OF TABLES

Table 3.1 ML models and training hyper-parameters for React-DeepONet and RecNet. 37

LIST OF FIGURES

Figure 2.1	Architecture of the basic DeepONet by Lu et al. (1)	16
Figure 2.2	Reconstruction of the left-over scalars from Representative scalars using ANNs	18
Figure 2.3	Workflow: Integrating React-DeepONet into PeleLMex Chemistry Loop	23
Figure 2.4	Operator-splitting approach of PeleLMex. Reaction ODEs integration can either be done by CVODE or by DeepONets (which uses Python-C API and JAX for inference).	24
Figure 3.1	Architecture of the React-DeepONet model (2)	28
Figure 3.2	Computational domain and boundary conditions for the backward-facing step configuration.	30
Figure 3.3	Initial temperature distribution showing the hot region downstream of the step.	30
Figure 3.4	Workflow: Training of the React-DeepONet and the RecNet frameworks	31
Figure 3.5	Snapshot images at different time steps used for sampling thermochemical states.	32
Figure 3.6	Thermochemical space comparison: (a) Full data space, (b) Sampled data space, (c) Comparison in $T-H_2O$ space.	33
Figure 3.7	Loss history for both React-DeepONet models under Pre-Training and Refined-Training regimes with increasing auto-regressive steps.	35
Figure 3.8	Separation of thermochemical space for the two React-DeepONet models in the $T-H_2O$ space	36
Figure 3.9	React-DeepONet offline predictions: initial conditions sampled from the 1300–1400 K temperature range. Solid line: Cantera solution; symbols: React-DeepONet.	38
Figure 3.10	React-DeepONet offline predictions: initial conditions sampled from the 1500–1600 K temperature range. Solid line: Cantera solution; symbols: React-DeepONet.	39
Figure 3.11	Comparison of CVODE and React-DeepONet predictions for key thermochemical variables at $t = 0.5$ ms in the Backward-Facing Step configuration. Each subplot shows results from detailed integration (CVODE), React-DeepONet predictions, and the corresponding error field.	41
Figure 3.12	Comparison of CVODE and React-DeepONet predictions for key thermochemical variables at $t = 1$ ms in the Backward-Facing Step configuration. Each subplot shows results from detailed integration (CVODE), React-DeepONet predictions, and the corresponding error field.	43
Figure 3.13	Comparison of CVODE and React-DeepONet predictions for key thermochemical variables at $t = 1.5$ ms in the Backward-Facing Step configuration. Each subplot shows results from detailed integration (CVODE), React-DeepONet predictions, and the corresponding error field.	45

Figure 3.14	Comparison of CVODE and React-DeepONet predictions for key thermochemical variables at $t = 1.75$ ms in the Backward-Facing Step configuration. Each subplot shows results from detailed integration (CVODE), React-DeepONet predictions, and the corresponding error field.	46
Figure 3.15	Comparison of CVODE and React-DeepONet predictions for key thermochemical variables at $t = 2$ ms in the Backward-Facing Step configuration. Each subplot shows results from detailed integration (CVODE), React-DeepONet predictions, and the corresponding error field.	47
Figure 3.16	Comparison of CVODE and React-DeepONet predictions for key thermochemical variables at $t = 2.25$ ms in the Backward-Facing Step configuration. Each subplot shows results from detailed integration (CVODE), React-DeepONet predictions, and the corresponding error field.	49
Figure 3.17	Comparison of CVODE and React-DeepONet predictions for key thermochemical variables at $t = 2.5$ ms in the Backward-Facing Step configuration. Each subplot shows results from detailed integration (CVODE), React-DeepONet predictions, and the corresponding error field.	50
Figure 3.18	Comparison of CVODE and React-DeepONet predictions for key thermochemical variables at $t = 2.75$ ms in the Backward-Facing Step configuration. Each subplot shows results from detailed integration (CVODE), React-DeepONet predictions, and the corresponding error field.	52

CHAPTER

1

INTRODUCTION

1.1 Motivation

Engineering simulations play a crucial role in modern technological advancements, enabling engineers and scientists to analyze and predict complex physical phenomena without the need for expensive and time-consuming experiments. These simulations provide a deeper understanding of fluid flow, heat transfer, structural mechanics, and electromagnetics, among other disciplines. By solving governing equations numerically, engineers can optimize designs, improve efficiency, and enhance the safety of engineering systems. High-fidelity computational models have become indispensable in industries such as aerospace, automotive, energy, and biomedical engineering, where accurate predictions are essential for innovation and performance assessment.

Among various engineering simulations, combustion simulations hold particular significance due to their direct application in energy generation, propulsion, and industrial processes. Combustion is a chemically reactive flow process that involves the interaction of multiple species, heat release, and turbulence, making its numerical modeling highly complex. The ability to accurately simulate combustion processes is critical for designing efficient propulsion systems, reducing emissions, and developing next-generation energy solutions. Due to the nonlinear and stiff nature of chemical reaction mechanisms, capturing the intricate interplay between fluid dynamics and chemistry remains a challenging computational problem. Advanced numerical methods and high-performance computing (HPC) resources are required to achieve accurate and efficient combustion simulations, making them an essential area of research in computational fluid dynamics (CFD) (3; 4).

Despite their critical importance, combustion simulations are computationally expensive due to the inherent complexities of reactive flows (5). One of the primary challenges is the stiffness of chemical reaction mechanisms, where vastly different time scales govern the evolution of species concentrations and temperature (6). Additionally, the complex dynamics of combustion involve turbulence-chemistry interactions, requiring fine-scale resolution to accurately capture flame structures, ignition, and extinction phenomena. The presence of a vast number of chemical species and reactions further increases the computational burden, making simulations highly high-dimensional and difficult to solve efficiently (7). Achieving high-fidelity results demands extremely fine spatial and temporal resolutions, leading to significant computational costs. As a result, combustion simulations remain highly expensive and often impractical, limiting progress in the development and optimization of combustion-based technologies.

In recent years, machine learning (ML) has emerged as a powerful tool to address the challenges associated with combustion simulations, offering a promising approach to reduce computational costs while maintaining accuracy. One of the key advantages of ML is its ability to handle high-dimensional combustion problems, where traditional numerical solvers

struggle with the vast number of species and complex chemical interactions (8). By leveraging data-driven techniques, ML models can identify underlying patterns in combustion chemistry and fluid dynamics, enabling efficient surrogate modeling and reduced-order representations. In particular, physics-informed machine learning (PIML) integrates fundamental physical laws, such as conservation equations and reaction kinetics, into ML architectures, ensuring physically consistent predictions (9). These advancements allow for the development of ML-based models that accelerate reaction integration, reduce dimensionality, and improve computational efficiency, making high-fidelity combustion simulations more practical for engineering applications (10; 11).

1.2 Background and Challenges in Reaction Integration

Combustion simulations remain one of the most computationally demanding areas in fluid dynamics due to the stiff nature of chemical kinetics and the high dimensionality of reacting systems. Accurate modeling requires the resolution of a wide range of chemical time scales, often differing by several orders of magnitude, which necessitates the use of implicit, adaptive ODE solvers such as CVODE (12). These solvers, while robust, are computationally expensive, especially when coupled with large chemical mechanisms consisting of hundreds of species and thousands of reactions.

To address this cost, various approaches have been proposed over the years. One class of methods focuses on reducing the size and complexity of the chemical mechanism. Techniques such as in situ adaptive tabulation (ISAT) and PRISM dynamically reuse previously computed results to avoid redundant integrations and accelerate simulations without significant loss in accuracy. Manifold-based approaches like the intrinsic low-dimensional manifold (ILDM) and computational singular perturbation (CSP) methods reduce the dimensionality of the chemical system by identifying slow invariant manifolds within the composition space (13; 14; 15; 16).

In parallel, data-driven approaches have emerged to reduce the dimensionality of the chem-

ical composition space. For example, principal component analysis (PCA) has been widely applied to identify and transport a reduced set of representative variables. These approaches assume that the full thermochemical state lies on a lower-dimensional manifold that can be effectively captured by a small number of principal components. Sutherland and Parente (17) introduced the transport of principal components (PC-transport) within DNS, while subsequent studies extended this to RANS and LES frameworks with promising accuracy and significant computational savings (18; 19).

More recently, machine learning (ML) techniques have been explored to represent the chemical source terms directly. These methods aim to learn the underlying mappings between thermochemical states and reaction rates or state evolution. For instance, neural networks have been used to approximate reaction rate functions, learn latent-space representations, or predict the state vector across time steps (20; 21). ChemNODE (22) is one such example that combines neural networks with ODE solvers to accelerate chemistry evaluation, while other studies have developed strategies to mitigate stiffness through learned neural ODEs (23). These ML-driven models offer a scalable and efficient alternative to traditional solvers, though challenges remain in ensuring physical consistency and stability during integration into existing CFD codes.

Despite these advancements, integrating ML-based surrogates into high-fidelity CFD solvers presents several challenges. These include ensuring compatibility with the solver architecture, maintaining thermodynamic consistency, and preserving computational efficiency when interfacing between C++ simulation frameworks and Python-based machine learning models. The present work contributes to this growing body of research by developing and integrating a DeepONet-based surrogate into PeleLMEx to accelerate the reaction step in operator-split reactive simulations.

1.3 Objective of This Work

The primary objective of this work is to develop and integrate a machine learning-based surrogate model for accelerating chemical kinetics in reactive flow simulations. In particular, the study focuses on replacing the traditional stiff ODE solvers used for reaction integration with a Deep Operator Network (DeepONet) framework (1). This surrogate is designed to predict the evolution of a reduced thermochemical state over time, enabling a more efficient update of species and temperature during the reaction step of operator-split CFD solvers.

By eliminating the need for numerically integrating stiff ODEs at every computational cell and time step, the proposed approach offers two major advantages. First, it enables the use of significantly larger time steps during the reaction phase, as the surrogate model directly learns the mapping between thermochemical states across time without the limitations imposed by stiffness. Second, the method reduces the overall computational cost of the simulation, which is especially impactful in large-scale 3D configurations and high-fidelity solvers like PeleLMex, where reaction integration often dominates total runtime (24).

This work further demonstrates the feasibility of integrating a Python-based DeepONet model into a complex C++ simulation framework by establishing a robust C++–Python interface. The successful integration paves the way for the practical application of data-driven methods in combustion modeling, offering an efficient and accurate alternative to traditional chemistry solvers.

1.4 Thesis Outline

In Chapter 2, the computational framework is introduced with a focus on PeleLMex, covering its modular structure, governing equations, and operator-splitting strategy. The chapter also details its reaction integration mechanisms. Following this, the Deep Operator Network (DeepONet) framework is discussed, including its architecture, dimensionality reduction using representative scalars, reconstruction of the full thermochemical state, training data gener-

ation, and model training methodology. The chapter concludes with an explanation of how DeepONet is integrated into PeleLMeX through a Python-C++ interface.

In Chapter 3, the results of this integration are presented. Simulations of the target problem are performed, and DeepONet-based predictions are compared against the conventional CVODE chemistry integration results. Accuracy and performance are evaluated at different time instances to assess the effectiveness of the surrogate model.

Finally, in Chapter 4, a summary of the work is provided along with the main conclusions. Potential directions for future research are outlined, with a focus on improving surrogate modeling techniques and extending their use in complex reacting flow simulations.

CHAPTER

2

METHODOLOGY

2.1 Overview of PeleLMeX and Its Reaction Integration Approach

PeleLMeX is an open-source high-fidelity computational fluid dynamics (CFD) code designed to solve the reacting Navier-Stokes equations in the low Mach number regime (25). It is a non-subcycling variant of its predecessor PeleLM and leverages adaptive mesh refinement (AMR) for computational efficiency. The code is implemented in C++ and employs MPI+X parallelization, where X represents an additional parallelization method such as OpenMP, CUDA, HIP, or SYCL. It supports block-structured AMR, making it suitable for complex combustion problems involving detailed chemical kinetics and transport phenomena. Additionally, PeleLMeX incorporates embedded boundary (EB) methods, allowing for accurate representation of complex

geometries.

2.1.1 PeleLMeX Submodules

PeleLMeX is a modular code that relies on multiple external and internal libraries to handle mesh adaptation, physics models, numerical solvers, and parallelization. The key submodules include:

AMReX (Adaptive Mesh Refinement Library)

- AMReX is the core framework that provides the adaptive mesh refinement (AMR) capability used by PeleLMeX (26).
- It enables dynamic grid refinement, where fine resolution is applied to regions of interest (e.g., reaction fronts, flame surfaces), while coarser grids cover less critical areas, significantly reducing computational cost.
- AMReX provides data structures (such as MultiFab and FabArray) for handling multi-level solutions and supports MPI-based domain decomposition for large-scale simulations.

PelePhysics (Combustion and Reaction Modeling)

- PelePhysics provides the chemical kinetics, transport properties, and thermodynamic models used in PeleLMeX (27).
- It includes:
 - Pre-tabulated thermodynamic data for various fuels and combustion environments.
 - Transport models (viscosity, thermal conductivity, and species diffusion), ensuring accurate treatment of reacting flows.
 - Detailed reaction mechanisms, including skeletal and reduced chemical kinetics.

- It interfaces with external libraries like Cantera and EQLib for accurate thermodynamic and transport property calculations.
- Supports various equations of state (EOS), including:
 - Ideal Gas Law (simplified, computationally efficient).
 - Real-Gas Equations of State (e.g., Soave-Redlich-Kwong (SRK) and Peng-Robinson) for high-pressure combustion simulations.

AMReX-Hydro (Hydrodynamics Module)

- AMReX-Hydro is a fluid dynamics package that PeleLMEx uses to solve reacting flow transport equations.
- Features include:
 - Second-order Godunov schemes for capturing flow features accurately.
 - Higher-order slope limiters to prevent numerical oscillations.
 - Low Mach number projection solvers to enforce divergence constraints and couple pressure-velocity dynamics.
- It also interfaces with PelePhysics to ensure consistent transport flux calculations.

SUNDIALS (Suite of Nonlinear and Differential/Algebraic Equation Solvers)

- PeleLMEx integrates SUNDIALS CVODE, a high-performance implicit ODE solver designed for stiff chemical kinetics (12).
- Reaction source terms are stiff, meaning that explicit integration would require prohibitively small time steps.
- CVODE employs an implicit backward differentiation formula (BDF) method, enabling stable integration of fast and slow reaction timescales within PeleLMEx.

- This allows efficient coupling of chemical kinetics with flow dynamics, ensuring accurate and computationally feasible reaction integration.

2.1.2 Governing Equations for Low Mach Number Flow

PeleLMeX solves the *low Mach number* form of the reacting Navier-Stokes equations (28). In this framework, the pressure is decomposed as:

$$p(\mathbf{x}, t) = p_0(t) + \pi(\mathbf{x}, t), \quad (2.1)$$

where $p_0(t)$ is a spatially uniform thermodynamic pressure, and $\pi(\mathbf{x}, t)$ is the perturbational pressure that drives the flow. The governing equations are:

Momentum Equation (Including advection, diffusion, and external forces):

$$\frac{\partial(\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u} + \tau) = -\nabla \pi + S_{\text{ext}, \rho \mathbf{u}}, \quad (2.2)$$

where ρ is the density, \mathbf{u} is the velocity, τ is the stress tensor, and $S_{\text{ext}, \rho \mathbf{u}}$ represents user-defined forcing terms.

Species Transport Equation:

$$\frac{\partial(\rho Y_m)}{\partial t} + \nabla \cdot (\rho Y_m \mathbf{u} + \mathcal{F}_m) = \rho \dot{\omega}_m + S_{\text{ext}, \rho Y_m}, \quad (2.3)$$

where Y_m is the mass fraction of species m , $\dot{\omega}_m$ is the molar production rate due to chemical reactions, and \mathcal{F}_m represents species diffusion fluxes.

Total Energy Equation:

$$\frac{\partial(\rho h)}{\partial t} + \nabla \cdot (\rho h \mathbf{u} + \mathcal{Q}) = S_{\text{ext}, \rho h}, \quad (2.4)$$

where h is the mass-weighted enthalpy, \mathcal{Q} is the heat flux, and $S_{\text{ext},\rho h}$ represents external heat sources.

Equation of State (EOS): PeleLMeX supports multiple *equations of state* (EOS), allowing the user to model different gas behavior depending on the application. For simplicity, the *ideal gas law* is often used, which relates pressure, density, and temperature as:

$$p_0(\rho, Y_m, T) = \frac{\rho RT}{W} = \rho RT \sum_m \frac{Y_m}{W_m}, \quad (2.5)$$

where W_m and W are the species and mean molecular weights, respectively, and R is the universal gas constant.

While the ideal gas law provides a *computationally efficient approximation*, PeleLMeX also allows for *real-gas models* such as the **Soave-Redlich-Kwong (SRK) equation of state** or **Peng-Robinson EOS**, which are more accurate for high-pressure combustion or non-ideal gas behavior. Users can select different EOS models depending on the problem setup.

The divergence constraint ensures that the velocity field is *solenoidal*, maintaining consistency with the low Mach number assumption. This leads to the constraint equation:

$$\nabla \cdot \mathbf{u} = \frac{1}{p_0 c_p T} (-\nabla \cdot \mathcal{Q} + S_{\text{ext},\rho h}) + \sum_m \frac{W}{\rho W_m} \frac{h_m}{p_0 c_p T} (-\nabla \cdot \mathcal{F}_m + \rho \dot{\omega}_m + S_{\text{ext},\rho Y_m}). \quad (2.6)$$

This equation links the evolution of temperature, chemical reactions, and velocity field, forming a *differential-algebraic equation (DAE) system* that PeleLMeX solves iteratively.

2.1.3 Reaction Integration Approach in PeleLMeX

PeleLMeX employs an operator splitting strategy to handle the complex and computationally intensive task of simulating chemically reacting flows. In such problems, multiple physical processes—such as advection, diffusion, and chemical reactions—occur on different timescales and exhibit varying degrees of stiffness. Applying a single time integration method to the fully

coupled system can be inefficient or unstable. Operator splitting addresses this by separating the evolution of physical processes and applying specialized numerical methods to each component.

In the context of reacting flow simulations, operator splitting enables the transport terms (advection and diffusion) to be treated separately from the reaction source terms. While transport is advanced using explicit or semi-implicit schemes, the chemical reactions are integrated using dedicated ODE solvers. Within PeleLMeX, this results in a set of spatially decoupled ODEs, one at each computational cell, where the species mass fractions and temperature evolve due to local reactions over a given fluid time step.

For the reaction step, PeleLMeX leverages the SUNDIALS library—specifically the CVODE package—for integrating the stiff ODE systems that arise from detailed chemical kinetics. CVODE supports both stiff and non-stiff ODEs using adaptive multistep methods and provides options for implicit integration, which is essential for handling the strong stiffness found in combustion reactions. Although SUNDIALS includes several integrators (such as ARKODE for additive Runge-Kutta methods or KINSOL for nonlinear solvers), CVODE is particularly well-suited for PeleLMeX due to its efficiency, robustness, and support for adaptive time stepping and Jacobian-based methods.

In PeleLMeX, the governing equations can be expressed in an additively partitioned form:

$$\frac{\partial \mathbf{U}}{\partial t} = \mathbf{F} + \mathbf{R}, \quad (2.7)$$

where \mathbf{U} is the state vector, \mathbf{F} includes all non-reactive contributions (advection, diffusion, etc.), and \mathbf{R} denotes the reaction source terms. The reaction term \mathbf{R} is evaluated and integrated separately over each fluid time step.

To ensure second-order temporal accuracy and consistency between the operator-split terms, PeleLMeX uses a lagged approximation algorithm. Initially, a provisional estimate of \mathbf{R} is used to compute \mathbf{F} , and then the reactions are integrated using CVODE with this fixed transport

forcing. After the reaction step, an updated \mathbf{R} is computed and the process is iterated. This outer iteration continues until the changes in \mathbf{F} and \mathbf{R} fall below a user-specified convergence tolerance. During each ODE integration, the transport term \mathbf{F} is held constant, allowing the reaction integration to be carried out independently and efficiently in each cell (24).

This splitting strategy, combined with the flexibility and performance of CVODE, allows PeleLMeX to accurately and efficiently evolve stiff chemical source terms, even when long time steps are used for the fluid advance. While the cost of reaction integration can dominate the overall runtime—particularly due to the stiffness and nonlinearity of the chemical system—the modular and scalable implementation in PeleLMeX makes it well suited for deployment on modern CPU and GPU architectures.

2.2 Deep Operator Networks (DeepONets)

Deep Operator Networks (DeepONets) are a class of machine learning models designed to approximate nonlinear operators that map functions to functions. Unlike conventional neural networks that learn finite-dimensional input-output mappings, DeepONets are capable of learning mappings between infinite-dimensional spaces, making them highly effective for problems governed by partial differential equations and other operator-based frameworks. Among the various machine learning architectures proposed in recent years for scientific computing tasks, DeepONets have demonstrated particular efficiency in learning continuous nonlinear operators, especially in domains with complex, multiscale behavior such as combustion processes (29; 30). Their generalizability and compact representation of dynamics make them a promising tool for accelerating scientific simulations (2).

In this work, DeepONets are used to approximate the solution of stiff ordinary differential equations that arise in combustion chemistry integration. The integration of chemical kinetics is often the most computationally expensive part of reactive flow simulations due to its multiscale nature. The oxidation of complex fuels involves hundreds to thousands of

chemical species and an even greater number of elementary reactions, leading to highly stiff and nonlinear systems. These stiffness characteristics pose significant challenges for traditional integration schemes, often requiring implicit solvers with adaptive time-stepping strategies that significantly increase computational cost. DeepONets offer a compelling alternative by directly learning the time-evolved mapping from input thermochemical states to future states, thus bypassing the need for conventional ODE solvers (27).

To enable this approach, an in-house DeepONet framework has been developed by the Computational Combustion and Energy Sciences Lab at North Carolina State University. This DeepONet is designed to learn the time evolution of a thermochemical state vector, effectively mapping a given input vector of species mass fractions, internal energy, and other relevant scalars to the updated state at a later time. By embedding the physical dynamics into the learning architecture, the network captures the temporal behavior of the underlying chemical kinetics without explicitly solving the governing equations at runtime. This approach enables significant computational savings while maintaining high fidelity in predicting species and temperature evolution in reactive flows.

2.2.1 Methodology

Deep Operator Networks (DeepONets) are powerful machine learning architectures developed to learn mappings between infinite-dimensional function spaces. Their flexibility allows for a wide range of applications, including solving differential equations, modeling multiscale physical systems, and, in our case, accelerating combustion chemistry integration. DeepONets are particularly well-suited for tasks involving stiff and nonlinear dynamics, such as chemical kinetics in reacting flows.

The fundamental idea behind DeepONets is to approximate an operator \mathcal{G} that maps an input function u to an output function $\mathcal{G}(u)$ (1). This mapping can then be evaluated at any desired value of an independent variable y , producing $\mathcal{G}(u)(y)$. In the context of combustion chemistry, the function u corresponds to a vector of thermochemical scalars—such as species

mass fractions and internal energy—at a given initial time. The variable y represents the time increment, and the output $\mathcal{G}(u)(y)$ corresponds to the predicted thermochemical state at the future time step.

As illustrated in Fig. 2.1, the DeepONet architecture consists of two coupled sub-networks: the *Branch Net* and the *Trunk Net*. The Branch Net takes as input the values of u sampled at a fixed number of sensor locations (in our case, a single sensor at a given time, so $\xi = 1$), and produces an output vector $[b_1, b_2, \dots, b_q]$. Meanwhile, the Trunk Net receives the target variable y (the time increment) and outputs another vector $[t_1, t_2, \dots, t_q]$. The final output of the DeepONet is computed by taking the dot product of these two vectors:

$$\mathcal{G}(u)(y) \approx \sum_{k=1}^q b_k(u) \cdot t_k(y)$$

This architecture is inspired by the universal approximation theorem for operators and has been shown to approximate both implicit and explicit operators with high accuracy (1; 31; 32).

In our application, we employ an in-house DeepONet framework developed at the Computational Combustion and Energy Sciences Lab at NC State. This network, referred to as *React-DeepONet*, is specifically designed to model the temporal evolution of thermochemical states in combustion (33; 2). The network is trained to predict the solution vector of representative thermochemical scalars after a small time increment, given the current state. These representative scalars form a reduced set from which the full species composition can be reconstructed, enabling reduced computational complexity while preserving accuracy.

React-DeepONet includes several enhancements over the standard DeepONet:

- **Solution Mapping Approach:** The network is trained to map current thermochemical states to future states over small, adaptive time steps, tailored for integration within reactive flow solvers.
- **Time Window Training:** To account for the sharp gradients and variability in chemical profiles, training is performed on time windows. Intermediate solutions are used as

additional initial conditions, effectively augmenting the training data.

- **Additional Input Sub-Network:** The model accommodates variability in thermodynamic parameters such as pressure, equivalence ratio, or dilution by including an auxiliary sub-network that processes these as additional inputs.
- **Latent Space Architecture:** For systems with large reaction mechanisms, a variant called *LS-React-DeepONet* introduces an autoencoder to reduce the dimensionality of the representative scalars. The DeepONet operates in the reduced latent space, and a decoder reconstructs the full solution. This significantly reduces network size and memory usage, improving training and inference efficiency for complex fuels.

These modifications make the React-DeepONet framework well-suited for high-fidelity yet computationally efficient chemistry integration in combustion simulations.

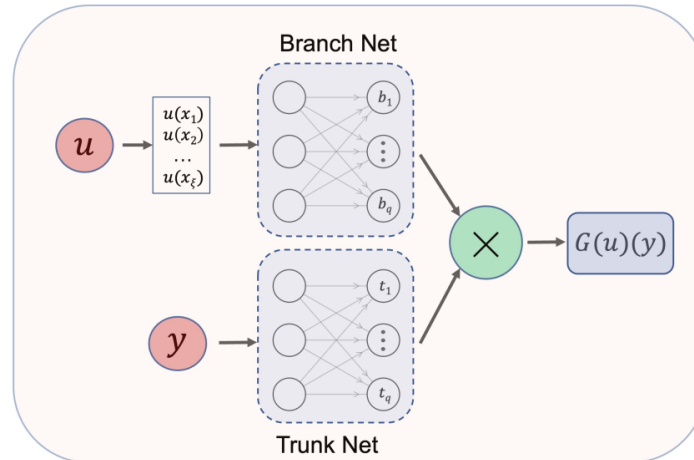


Figure 2.1: Architecture of the basic DeepONet by Lu et al. (1)

2.2.2 Reduction of the solution vector of thermochemical state using representative scalars

Combustion chemistry models often involve a large number of species and corresponding thermochemical variables. This high dimensionality significantly increases the computational cost of simulating reactive flows. To address this, the React-DeepONet framework uses a reduced representation of the thermochemical state, referred to as *representative scalars* (34). These include a carefully selected subset of species and temperature, from which the full state can be recovered through post-processing.

The selection of representative scalars is guided by a data-driven dimensionality reduction technique. Specifically, principal component analysis (PCA) is applied to the reaction rates obtained from detailed simulation data. This approach identifies species whose reaction activity plays a key role in determining the overall behavior of the chemical system. In essence, it captures the idea that certain species drive the evolution of others due to their central roles in reaction pathways.

The process begins by centering the reaction rate data to remove any bias due to mean behavior. A covariance matrix is then constructed to understand how the reaction rates of different species are correlated. PCA is performed on this matrix to extract principal components that describe the dominant modes of variability in the dataset. These components are ordered by the amount of variance they capture, and only the top components that explain the majority of the variance (typically 99%) are retained (34).

Once the most influential components are identified, the species that contribute the most to them are selected as representative species. Temperature is included as a necessary thermodynamic variable. To further refine the selection, the same process is repeated on the reduced subset to compress the state vector even further, while ensuring that essential information is preserved.

This two-stage PCA-based procedure, inspired by the work of Alqahtani and Echehki (34), enables a compact yet expressive representation of the chemical system. It allows React-DeepONet

to operate efficiently while maintaining predictive fidelity for complex fuels involving hundreds of interacting species.

2.2.3 Reconstruction of the remaining thermochemical scalars

In the React-DeepONet framework, only a selected subset of thermochemical scalars—referred to as representative scalars—is directly predicted by the network. The remaining species, which are not explicitly modeled, are reconstructed using regression techniques based on this resolved subset. In this study, artificial neural networks (ANNs) are used to learn the mapping between the representative scalars and the rest of the thermochemical state.

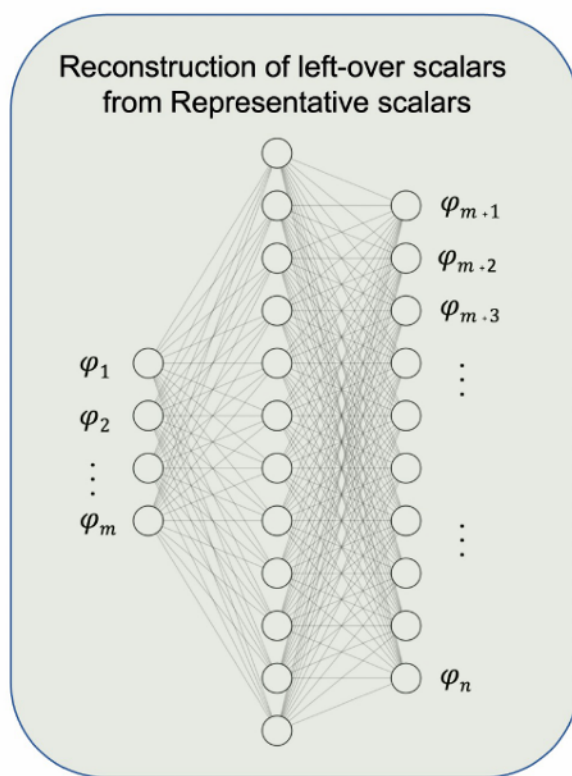


Figure 2.2: Reconstruction of the left-over scalars from Representative scalars using ANNs

These ANNs are trained using the same simulation data that was used for constructing the DeepONet model, ensuring consistency in chemical behavior across the framework. The

inputs to the ANNs are the representative scalars, and the outputs are the remaining species or thermochemical variables that complete the state vector.

The architecture of these reconstruction ANNs is intentionally shallow, typically comprising only two hidden layers. This design choice reflects the fact that the representative scalars are carefully chosen to retain the dominant chemical features, making them sufficient for accurately inferring the remaining species without requiring deep or complex network structures.

2.3 Training Data Generation and Model Training Mechanism

The React-DeepONet framework requires a large, diverse, and physically representative dataset for supervised training. To generate this dataset, a coarse-grid reacting flow simulation is performed using detailed chemical kinetics. This simulation captures the spatiotemporal evolution of temperature and species mass fractions under realistic flow and combustion conditions, forming the foundation for constructing the surrogate model's training data.

Thermochemical states are sampled from multiple time instances across the domain to ensure comprehensive coverage of the composition space. To avoid redundancy and ensure balanced representation, K-means clustering (35) is applied to the collected data. A fixed number of states are selected from each cluster, resulting in a reduced yet diverse dataset that reflects a broad spectrum of chemical regimes. Each sampled thermochemical state is then used as an initial condition for a zero-dimensional homogeneous reactor simulation. These simulations, conducted using Cantera (36), evolve the species and temperature profiles over time at constant pressure, matching the local thermodynamic conditions of the sampled states.

The resulting time series trajectories are used to train the DeepONet model. Training is implemented in two phases: Pre-Training and Refined-Training. During Pre-Training, each sampled state is evolved over a short time interval, and the model is trained to minimize the mean absolute error (MAE) between its prediction and the reference state after a small time increment. This phase serves to initialize the network weights and guide them toward

meaningful approximations of the underlying operator.

Refined-Training extends this approach through auto-regressive roll-outs. In this phase, the model’s own outputs are recursively used as inputs to forecast future states over multiple time steps. The accumulated prediction error is then minimized by comparing the forecasted states against the corresponding reference solutions. This training strategy improves the model’s long-term stability and accuracy, which are essential for deployment within time-dependent CFD solvers. To support this process, each sampled thermochemical state must be evolved over multiple time steps in the zero-dimensional simulations, matching the length of the roll-out horizon used during training.

Through this two-stage training pipeline—grounded in physically consistent data, robust sampling strategies, and sequential learning—the React-DeepONet model learns to accurately approximate the evolution of representative thermochemical scalars. This prepares it for integration as a surrogate chemistry solver in complex, high-fidelity reactive flow simulations.

2.4 DeepONet integration in PeleLMeX

2.4.1 Integration Challenges and Technical Constraints

In this study, a neural network surrogate model based on the Deep Operator Network (DeepONet) framework is proposed to accelerate reactive flow simulations by bypassing the expensive reaction integration step typically performed using stiff ODE solvers. In combustion simulations, the integration of detailed chemical kinetics can account for up to 70% of the total computational cost, especially when large chemical mechanisms are used. By replacing this step with a trained DeepONet that maps the thermochemical state forward in time, substantial speedups can be achieved. Furthermore, to reduce the dimensionality of the input and output vectors, the DeepONet is trained to predict only a subset of representative scalars, including temperature and selected species mass fractions. The remaining species are reconstructed through regression using the resolved subset, thus balancing computational efficiency and

predictive accuracy.

While the surrogate-based approach offers significant potential, its integration into a complex computational fluid dynamics (CFD) code like PeleLMeX poses several non-trivial challenges. The first major challenge lies in the compilation process. PeleLMeX uses a hybrid build system where CMake is employed to manage third-party dependencies such as SUN-DIALS, but the final compilation and linking of the main executable are governed by AMReX-provided Makefiles. Introducing Python and NumPy into this environment requires modifying these Makefiles to link the appropriate Python libraries and headers. In large repositories like PeleLMeX, which consist of thousands of source files and nested Makefiles across modules, even minor changes can lead to build errors or runtime instability. Ensuring a clean and functional build with Python integrated is particularly difficult given the interdependencies among modules.

A second critical challenge is the development of an efficient and lightweight interface between the C++ simulation code and the Python-based DeepONet model. Although using DeepONet avoids the computational burden of solving stiff ODEs, the benefit can be offset if the model inference becomes a new bottleneck. The Python C API and NumPy C API must be used with precision to avoid excess overhead during data exchange. Since the DeepONet prediction is invoked at every cell in the domain and at every time step, even small inefficiencies in the interface can lead to substantial cumulative slowdowns. Designing a minimal overhead communication scheme between AMReX's MultiFab data structures and NumPy arrays is essential to preserve the computational savings offered by the surrogate model.

Closely tied to the interfacing challenge is the third issue of memory management. Improper allocation or conversion of data structures between C++ and Python can lead to memory leaks or segmentation faults during long simulations. Special care must be taken to manage memory ownership between the simulation code and the Python interpreter, particularly when dealing with large batches of per-cell data in high-resolution simulations. Avoiding unnecessary deep copies and ensuring that memory is released correctly after each prediction step are vital to

maintaining the stability of the coupled system.

A fourth challenge arises from the need to ensure compatibility with parallel execution models. PeleLMeX is highly parallelized, relying on OpenMP and MPI for CPU execution and CUDA or HIP for GPU execution. Inserting Python prediction calls within a multi-threaded C++ environment requires careful attention to thread safety and interpreter state. Since the Python Global Interpreter Lock (GIL) can limit concurrent execution, steps must be taken to manage or isolate Python calls on a per-thread basis. In MPI-parallel runs, additional care must be taken to prevent conflicts when multiple processes interact with Python simultaneously.

Finally, the integration process must also address hardware constraints, especially in GPU-enabled builds. While PeleLMeX supports GPU execution, the current Python-based DeepONet model runs on CPU. This introduces the overhead of frequent data transfers between host and device memory, potentially negating the advantages of GPU acceleration. Ensuring that the surrogate model operates efficiently in such hybrid environments requires further architectural consideration and, in the long term, may necessitate porting the model itself to GPU-compatible frameworks.

2.4.2 Implementation Workflow and Runtime Integration

Integrating the Python-based DeepONet framework into the C++ infrastructure of PeleLMeX requires substantial effort and precise coordination between the build system, the reaction integration code, and the memory management mechanisms. This section outlines the step-by-step methodology adopted to embed the DeepONet model as a surrogate for chemistry integration in PeleLMeX, replacing the traditional stiff ODE solvers.

The first stage in the integration process involves modifying the build system to support Python-based inference at runtime. PeleLMeX is built upon a hybrid compilation system where core dependencies such as SUNDIALS are managed through CMake, but the final compilation and linking of the main executable is handled by AMReX-driven Makefiles. In this setup, it is critical to ensure that the Python and NumPy headers and associated linking flags are properly

included in the Makefiles of both PeleLMeX and its dependency PelePhysics (which houses the reaction integration modules). This inclusion enables the compiled executable to locate and link with the Python interpreter and NumPy C API during runtime, which is essential for invoking Python-based models.

Following the build system configuration, the next step involves modifying the source code of the selected reactor. PeleLMeX supports multiple chemistry integrators—such as CVODE, ARKODE, RK64, and KINSOL. Among these, the CVODE integrator is the most widely used due to its robust support for MPI parallelism, GPU acceleration, and efficient handling of stiff reaction systems. Therefore, the DeepONet integration was implemented within the CVODE reactor module of PelePhysics.

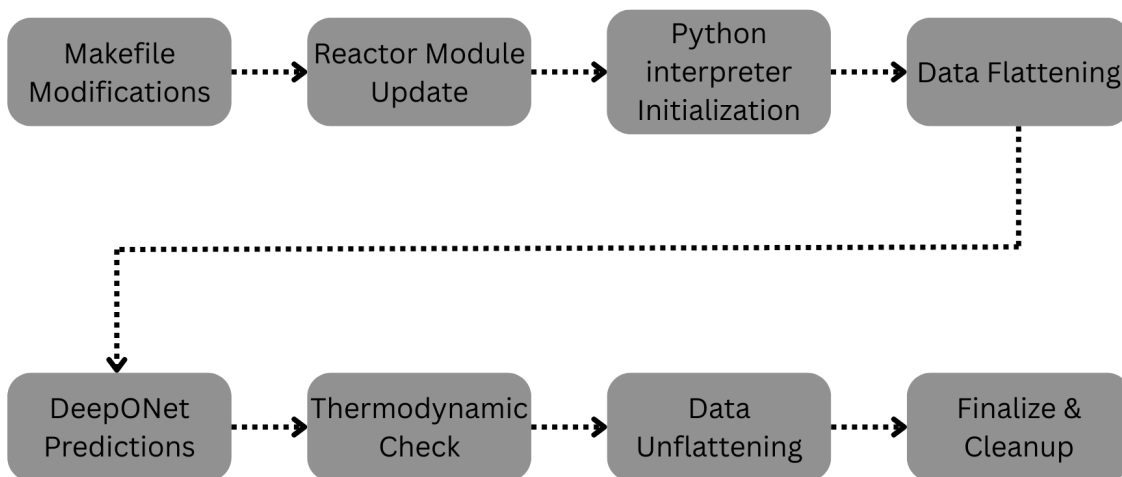


Figure 2.3: Workflow: Integrating React-DeepONet into PeleLMeX Chemistry Loop

Within the CVODE reactor code, a Python initialization routine is added. This routine sets up the Python interpreter, ensures the proper initialization of the NumPy module, and

imports the required DeepONet prediction module from the Python environment. The function also prepares the pointer to the specific Python function that will be called during reaction integration. This initialization ensures that all Python modules and dependencies are available before the start of the simulation.

Once the Python environment is initialized, the actual integration of DeepONet into the simulation loop takes place. In PeleLMEx, the reaction integration is performed cell-by-cell over localized regions of the computational domain, referred to as *boxes* in AMReX terminology. Each box is a chunk of the domain assigned to a compute task, and the simulation iterates over these boxes to perform updates on the contained cells.

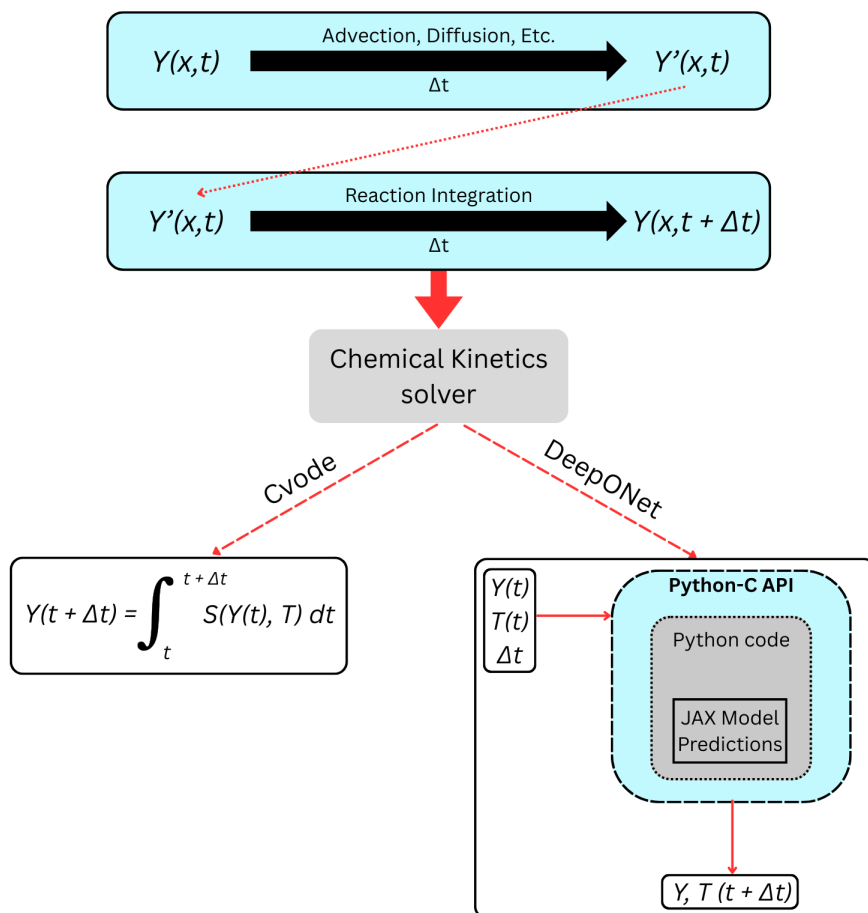


Figure 2.4: Operator-splitting approach of PeleLMEx. Reaction ODEs integration can either be done by CVODE or by DeepONets (which uses Python-C API and JAX for inference).

Inside the loop that traverses the cells within each box, the code performs a process called *data flattening*. This step involves extracting the necessary thermochemical quantities—such as species mass fractions, temperature, and internal energy—from AMReX’s native multidimensional Array4 data structures and converting them into one-dimensional arrays that are compatible with the Python interface. These flattened arrays are used to construct the input to the DeepONet model.

Traditionally, this is the point where the CVODE integrator would solve the stiff ODE system governing chemical kinetics for each cell. However, in the integrated version, this step is bypassed. Instead, the representative species and temperature values are assembled into input arrays. Additional quantities like internal energy and the time step size are also prepared. These values are passed to an intermediate interface function that bridges the C++ and Python environments using the Python C API and NumPy C API. This interface efficiently transfers data to the Python runtime, where the DeepONet model predicts the representative scalars, and an accompanying ANN-based regression model reconstructs the remaining species.

Once the full thermochemical state is predicted, the output is returned to the C++ side as a flat array. A thermodynamic consistency check is then enforced by passing the predicted temperature and internal energy to the equation-of-state (EOS) routines in PeleLMEx. This step ensures that the temperature prediction aligns with the internal energy constraints of the mixture, maintaining physical fidelity. After this, the one-dimensional output arrays are *unflattened* and converted back into the native Array4 format of AMReX, effectively updating the state variables in-place for each cell.

After the cell-level loop concludes, a finalization routine is called to properly release any Python-related resources. This function decrements Python reference counts for the loaded module and prediction function and frees the dynamically allocated memory used for the data exchange buffers. Although the Python interpreter itself is not finalized within this routine—since it may be reused later in the simulation—the cleanup ensures that no memory leaks or dangling pointers remain from the surrogate integration calls.

This entire integration process allows PeleLMeX to perform chemistry updates using a surrogate model rather than solving the governing ODEs explicitly. By doing so, significant reductions in computational cost are achieved without sacrificing the resolution of key species and temperature profiles essential for modeling reactive flows. The approach balances interoperability between C++ and Python while maintaining performance and stability within the AMReX-based simulation framework.

CHAPTER

3

RESULTS AND VALIDATION

3.1 Background

The proposed surrogate model for accelerating chemistry integration in PeleLM_X is built upon the React-DeepONet framework. Rather than solving the stiff system of ordinary differential equations (ODEs) arising from detailed chemical kinetics—typically handled through operator splitting—this approach replaces the reaction step with a learned operator that maps the current thermochemical state to its future evolution. Specifically, the model is trained to predict the updated values of a subset of thermochemical scalars, referred to as *representative scalars*, which include key species and temperature. The remaining species are reconstructed through post-processing, resulting in reduced computational complexity and significant speedup.

The React-DeepONet architecture utilized in this study, shown in Figure 3.1, consists of

two input pathways: a Branch Network and a Trunk Network. The Branch Network receives the current thermochemical state $\phi(t)$, while the Trunk Network takes the time increment δt as input. Each sub-network processes its respective input to generate a feature vector, and the final prediction at time $t + \delta t$ is computed as the inner product of these vectors. This architecture enables the network to learn a general operator that advances the state across arbitrary time steps without explicitly solving the governing ODEs.

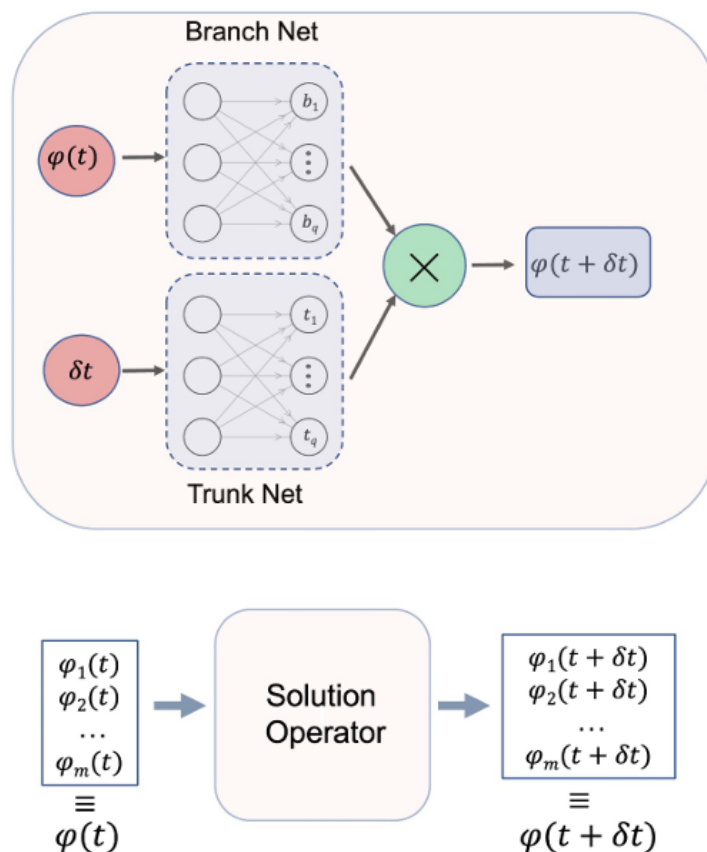


Figure 3.1: Architecture of the React-DeepONet model (2)

In this implementation, any valid thermochemical state $\phi(t)$ sampled from the training data can be used as an input condition to the DeepONet. This flexibility enhances the generalization capability of the model and allows it to respond accurately to a wide range of initial conditions

during simulation. The time increment δt , which can vary up to the simulation’s physical time step Δt , serves as the second input stream and helps the model adapt its prediction horizon accordingly.

By embedding the DeepONet surrogate directly into the PeleLMEx chemistry integration loop, the computational cost associated with stiff ODE solvers is substantially reduced. This data-driven operator model provides a flexible and efficient alternative that preserves the fidelity of combustion simulations while enabling faster time integration. The combined use of reduced state prediction and post-hoc reconstruction of the full thermochemical state offers further gains in efficiency, making this framework a practical solution for large-scale reactive flow computations.

3.2 Case Setup, Data Generation, and Model Training Summary

3.2.1 Simulation Configuration

The computational setup used for training and validation is based on a two-dimensional premixed methane–air flame anchored behind a backward-facing step. The simulation is performed on a 0.08×0.02 m² rectangular domain. The bottom-left and top-right corners of the domain are located at $(-0.01, -0.01)$ and $(0.07, 0.01)$, respectively. The base grid is uniformly discretized into 256×64 cells, and adaptive mesh refinement (AMR) is not activated during training data generation.

The flow is aligned with the x -direction. A Dirichlet (inflow) boundary condition is applied on the x -low boundary, while a Neumann (outflow) boundary condition is imposed on the x -high boundary. No-slip wall conditions are enforced on the top and bottom transverse boundaries. The simulation employs a Cartesian coordinate system.

The backward-facing step geometry is represented using the Embedded Boundary (EB)

method in PeleLMeX. The flame stabilizes in the recirculation zone formed downstream of the step. Figure 3.2 illustrates the domain layout and boundary conditions, while Figure 3.3 shows the initial temperature distribution.

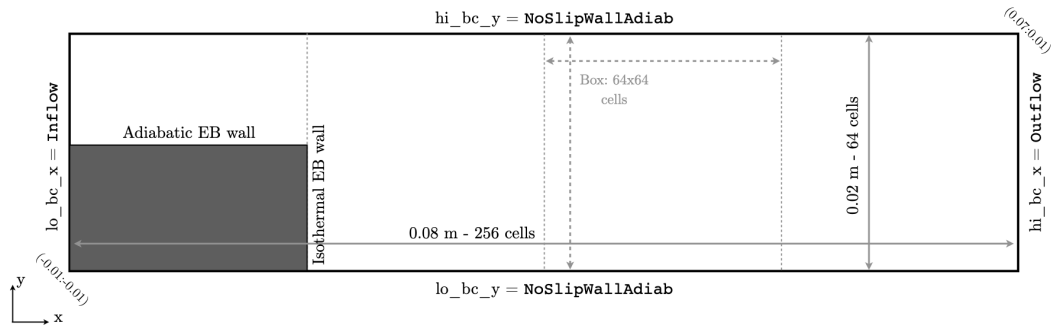


Figure 3.2: Computational domain and boundary conditions for the backward-facing step configuration.

The inlet and initial temperature of the gas is 298 K, and the operating pressure is 101325 Pa. The inlet mixture consists of methane with a mass fraction of 0.0445 and oxygen with a mass fraction of 0.223, with the remaining fraction made up of nitrogen. The region behind the step is initialized at a higher temperature of 1800 K, facilitating flame anchoring. The embedded boundary wall is maintained at a constant temperature of 300 K.

A mean inlet velocity of 10 m/s is specified at the left boundary to drive the premixed flow through the domain under low Mach number conditions.

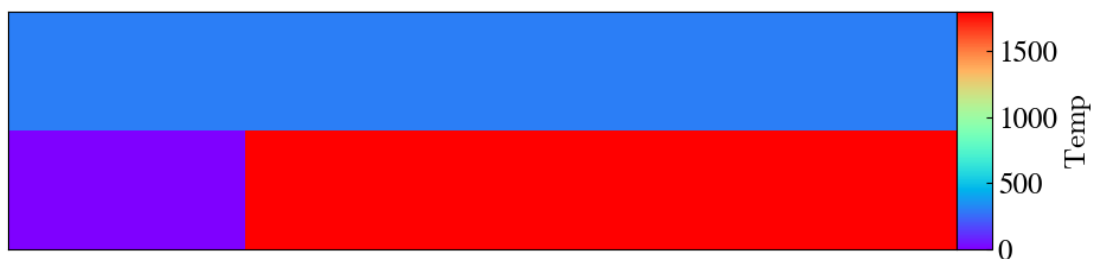


Figure 3.3: Initial temperature distribution showing the hot region downstream of the step.

3.2.2 Data Sampling and 0D Evolutions

The overall workflow for the training process for the React-DeepONet framework is shown in Fig. 3.4. To train the React-DeepONet model for this problem, thermochemical states are sampled from a coarse-grid simulation of the backward-facing step flame. Multiple snapshots from different time instances are recorded during the simulation to ensure that the sampled data reflects the dynamic evolution of the flame across both spatial and temporal domains. Representative snapshots from this coarse-grid simulation are shown in Figure 3.5, which serve as the source from which individual thermochemical states are extracted.

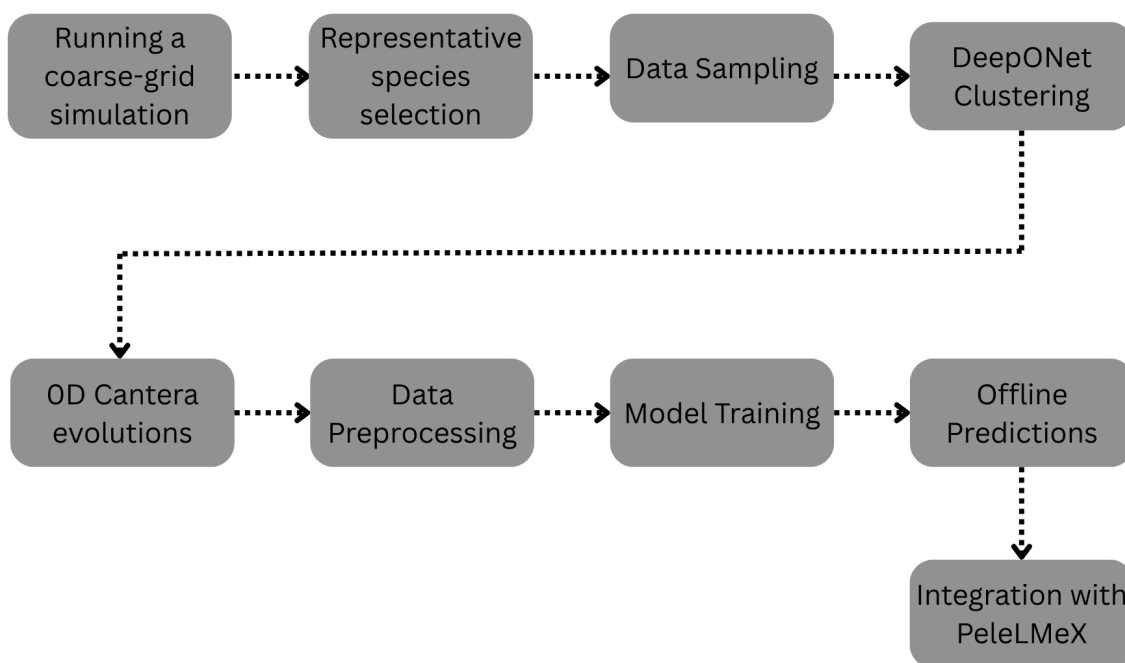


Figure 3.4: Workflow: Training of the React-DeepONet and the RecNet frameworks

To better capture the long-term evolution of chemical kinetics and to specialize the React-DeepONet model for particular subsets of the composition space, the sampled thermochemical space is divided into two distinct zones. A separate React-DeepONet model is trained for each zone, allowing for improved accuracy by tailoring each model to a narrower range of input

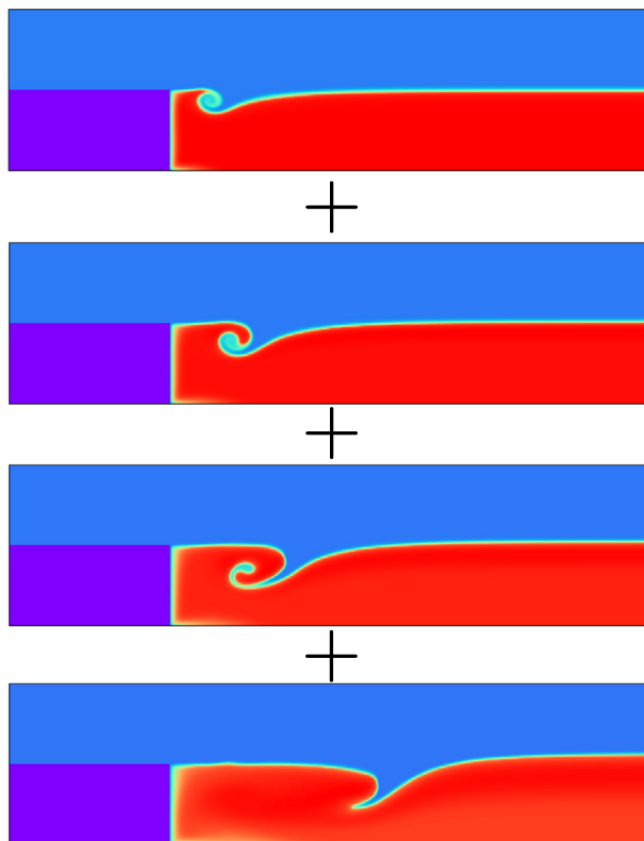


Figure 3.5: Snapshot images at different time steps used for sampling thermochemical states.

conditions. The zoning is based on the clustering behavior of species profiles and temperature in the sampled data, which helps distinguish between regions with differing reaction progress or chemical characteristics.

From the full thermochemical dataset, approximately 200,000 data points are randomly sampled to serve as initial conditions for zero-dimensional (0D) homogeneous reactor simulations. These simulations are performed using Cantera under constant-pressure conditions, with each point evolved for 400 time steps using a fixed time step of 2.0×10^{-6} seconds. These trajectories provide the sequential data required to train the DeepONet models in both the pre-training and refined-training phases.

To visualize the diversity and representativeness of the sampled states, the thermochemical space is analyzed before and after the sampling process. Figure 3.6 shows this comparison, where the full and sampled datasets are projected onto a two-dimensional space defined by

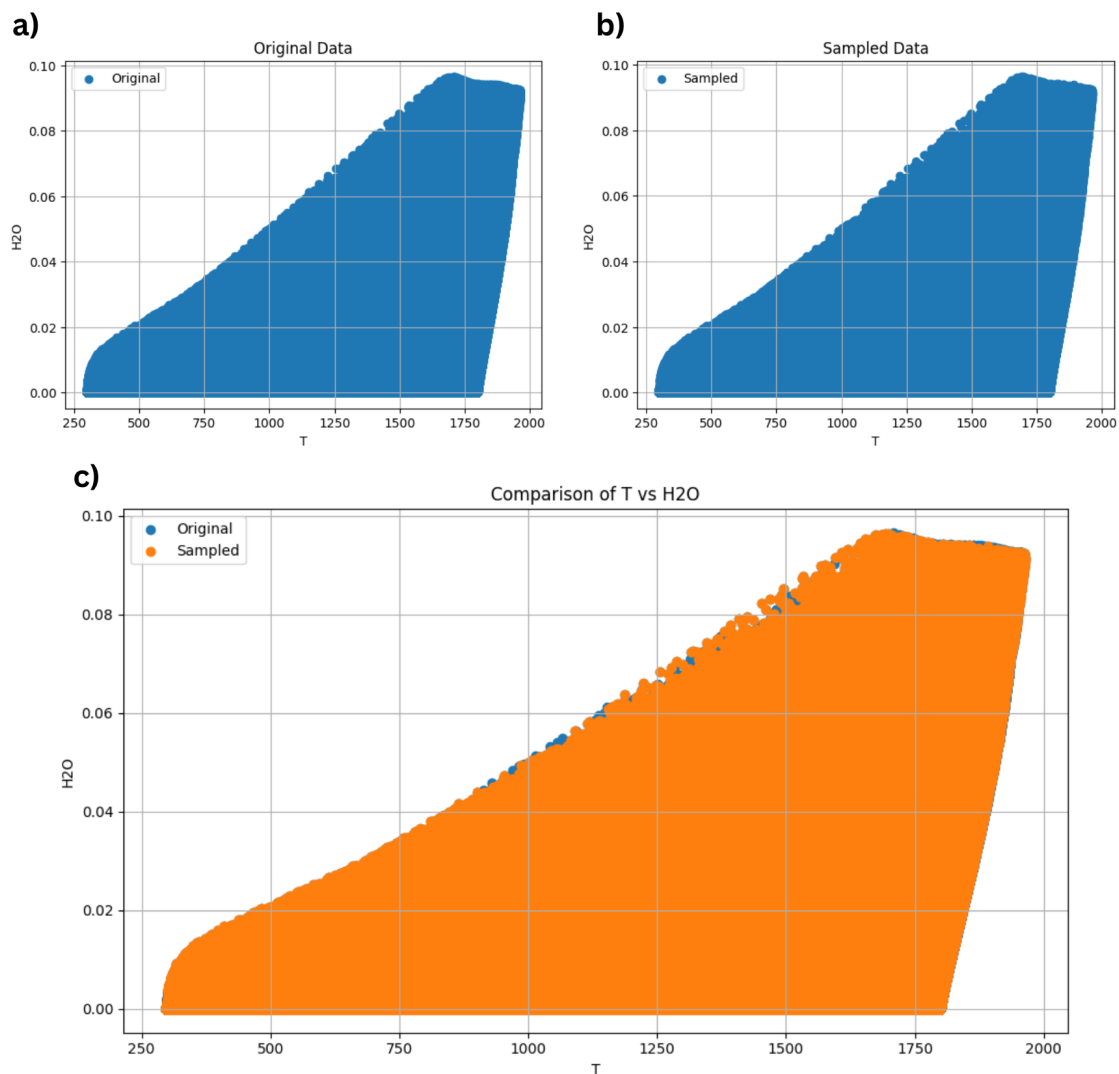


Figure 3.6: Thermochemical space comparison: (a) Full data space, (b) Sampled data space, (c) Comparison in T - H_2O space.

temperature and a key product species. The original dataset spans a wide range of thermal and compositional states, while the sampled data preserves the essential features of the original space, indicating a well-distributed and diverse sampling procedure. This ensures that the React-DeepONet models are trained on physically meaningful and representative input states, enabling robust generalization to unseen data during simulation.

3.2.3 Model Training and Convergence Behavior

All machine learning models used in this study are implemented using the JAX framework and trained on an NVIDIA RTX A6000 GPU. The Adamax optimizer is employed for gradient-based optimization, with model-specific parameters and training hyperparameters listed in Table 3.1.

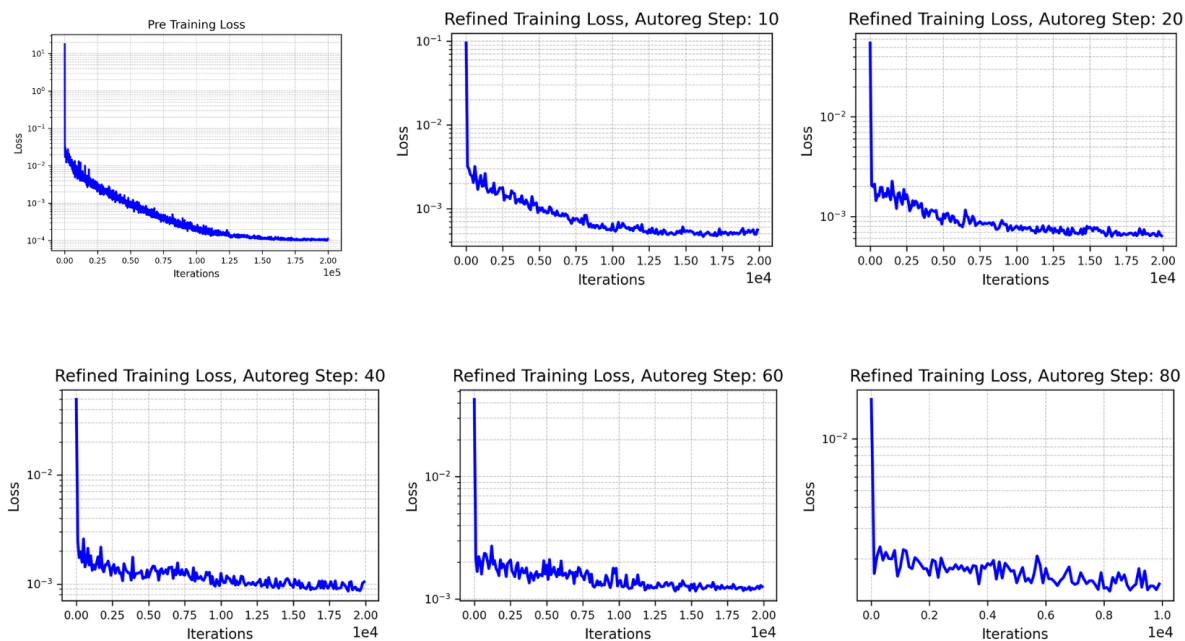
The training process is divided into two regimes: *Pre-Training* and *Refined-Training*. As the name suggests, Pre-Training is conducted first and serves as a preconditioner for Refined-Training. During this phase, each sampled thermochemical state is used to predict the state at a small time increment, and the model is trained to minimize the mean absolute error (MAE) between predicted and reference outputs. This phase helps guide the model parameters towards physically consistent initialization.

Refined-Training is then carried out in an auto-regressive manner, where the model's prediction at each time step is recursively used as input to forecast the next state. Instead of training directly for a large number of auto-regressive steps, the rollout horizon is progressively increased (e.g., 10, 20, etc.). This gradual strategy stabilizes the training process and ensures that the model retains predictive accuracy over extended time intervals. At each stage, the loss is accumulated over the entire trajectory and backpropagated to update the model parameters.

Both React-DeepONet models—each corresponding to a specific thermochemical region (as shown in fig. 3.8)—are trained to convergence for all rollout stages. As expected, the final saturated loss increases with longer auto-regressive steps, since the error compounds over time. Nonetheless, the training remains stable, and the model demonstrates consistent convergence across both training regimes.

The evolution of training losses for both models under Pre-Training and Refined-Training is shown in Fig. 3.7, highlighting the stability and effectiveness of the two-phase training approach.

First React-DeepONet Losses



Second React-DeepONet Losses

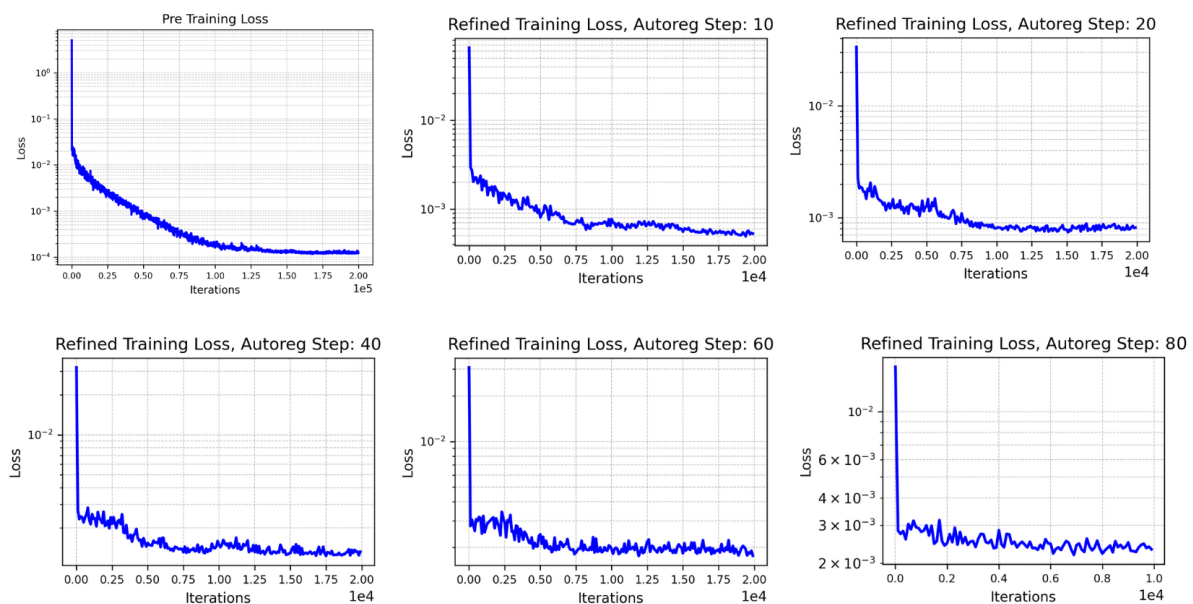


Figure 3.7: Loss history for both React-DeepONet models under Pre-Training and Refined-Training regimes with increasing auto-regressive steps.

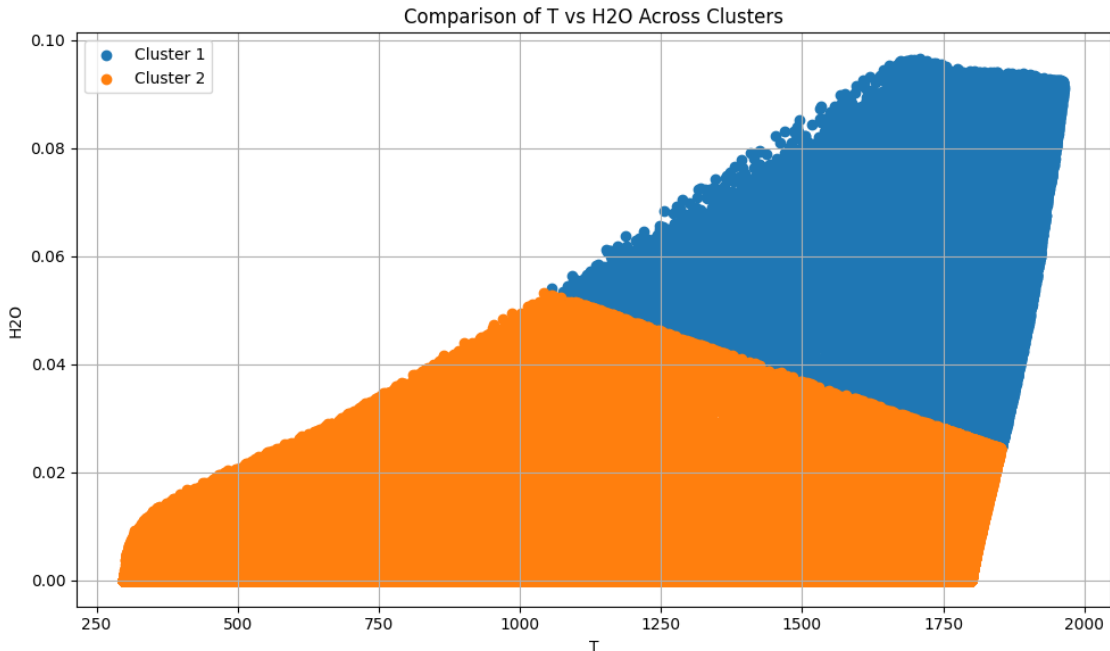


Figure 3.8: Separation of thermochemical space for the two React-DeepONet models in the T - H_2O space

3.2.4 Offline Predictions

Before integrating the trained React-DeepONet models into the PeleLMEx solver, their predictive capability is rigorously evaluated in an offline setting. The goal of this step is to ensure that the models can reliably capture the underlying dynamics of chemical kinetics over multiple time steps when operated in an auto-regressive fashion.

In this mode, the trained React-DeepONet model is provided with an initial thermochemical state, and it sequentially predicts the state at the next time step using its own previous prediction as the new input. This continues over a fixed prediction horizon to produce full 0D solution trajectories. These predicted trajectories are then compared with the corresponding ground-truth solutions obtained from zero-dimensional Cantera simulations, which serve as the high-fidelity baseline.

The initial conditions for this evaluation are randomly sampled from the same thermochemical space used during training. For each sampled condition, a full trajectory is generated

Table 3.1: ML models and training hyper-parameters for React-DeepONet and RecNet.

Parameters	React-DeepONet (1 st & 2 nd)	RecNet
Architecture	Branch Net – [7,100,100,100,100,420] Trunk Net – [1,100,100,100,100,420]	[7,50,50,50,50,13]
Total Parameters	148.4 K	9513
Loss Formulation	MAE	MAE
Training Iterations	2.0×10^5	6×10^5
Refined Training Iterations	0.9×10^5	-
Mini-Batch Size	10,000	25,000
Learning Rate	5×10^{-2}	2×10^{-2}
Refined Training Mini-Batch Size	6,000	-
Refined Training Learning Rate	3×10^{-3}	-

using both Cantera and the React-DeepONet model. As shown in Figures 3.9 and 3.10, the model demonstrates strong agreement with the reference solutions, accurately capturing the transient evolution of temperature and key species mass fractions across various initial states. Figure 3.9 corresponds to initial temperatures in the range of 1300–1400 K, while Figure 3.10 shows predictions for initial temperatures between 1500–1600 K. In both cases, solid lines represent the Cantera ground truth, and symbols indicate React-DeepONet predictions.

These results highlight the effectiveness of the pre-training and refined-training methodology—especially the progressive increase of auto-regressive steps—in ensuring stability and accuracy across extended prediction horizons.

In addition to the React-DeepONet, a shallow feedforward neural network—referred to as *RecNet*—is trained to reconstruct the mass fractions of non-representative species based on the output of the React-DeepONet. This reconstruction step is critical for restoring the full thermochemical state vector required by the CFD solver. The RecNet is trained separately using the same Cantera data and shows high accuracy in capturing the nonlinear correlations between representative and non-representative species.

Together, these models form a complete surrogate framework for chemistry integration that balances computational efficiency with high-fidelity predictions, setting the foundation for deployment in full-scale reactive flow simulations.

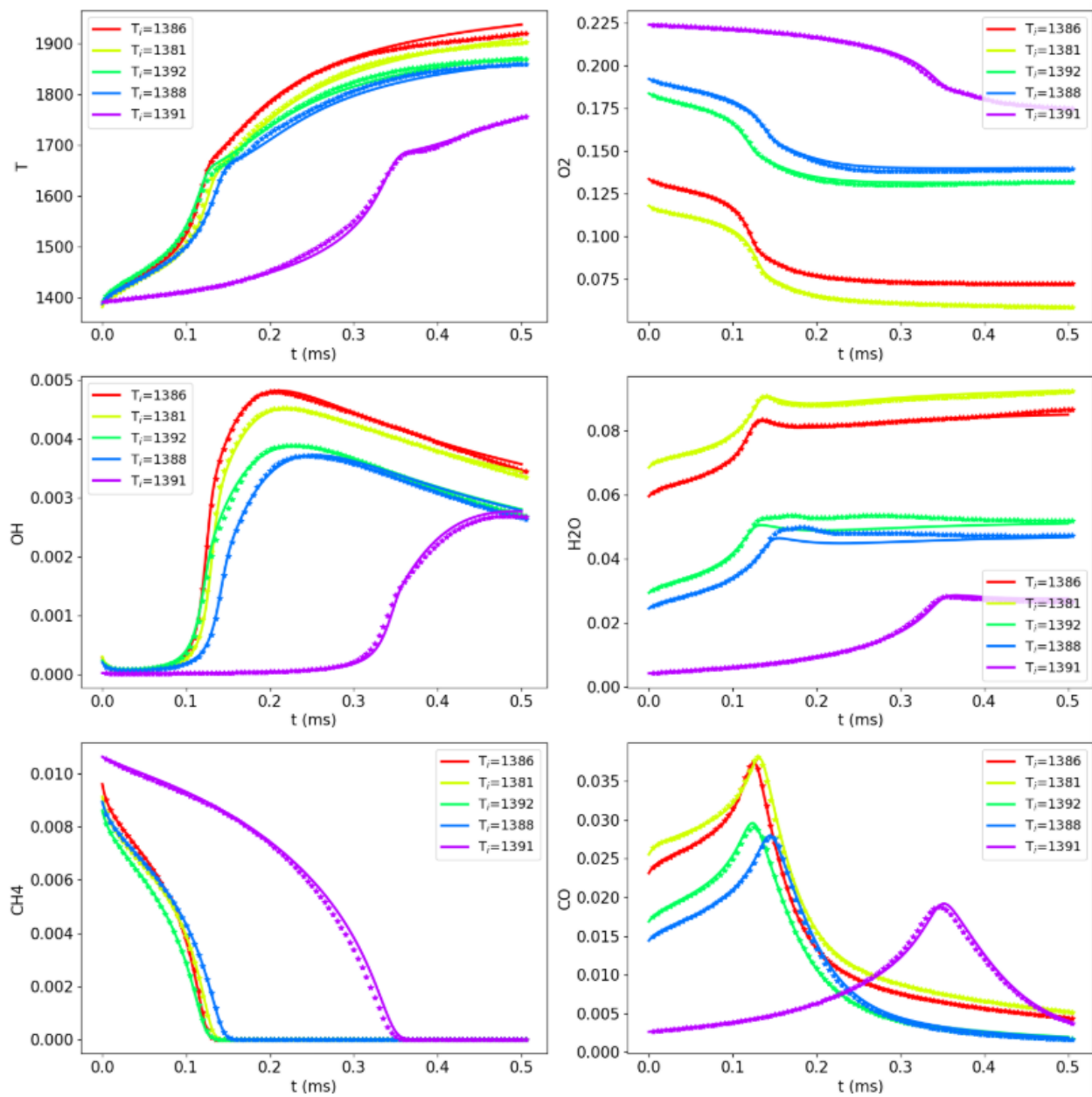


Figure 3.9: React-DeepONet offline predictions: initial conditions sampled from the 1300–1400 K temperature range. Solid line: Cantera solution; symbols: React-DeepONet.

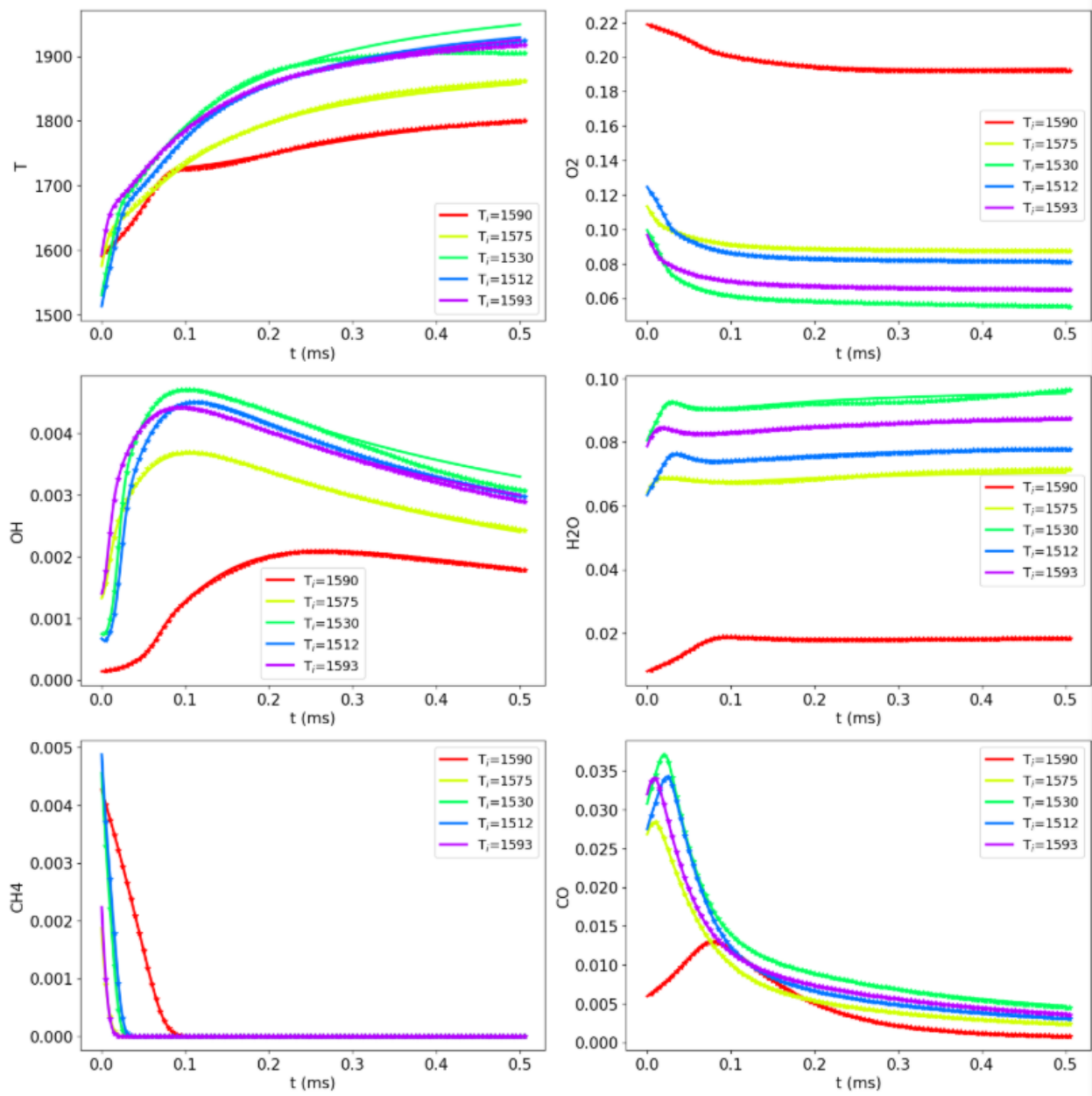


Figure 3.10: React-DeepONet offline predictions: initial conditions sampled from the 1500–1600 K temperature range. Solid line: Cantera solution; symbols: React-DeepONet.

3.3 Results

This section presents the performance of the integrated surrogate framework in the backward-facing step (BSF) simulation. The goal is to evaluate the accuracy of DeepONet-based predic-

tions against the standard CVODE solver and assess the resulting computational benefits.

3.3.1 Scalar Field Comparisons

After successful training and validation, the React-DeepONet and RecNet models are integrated into the PeleLMex solver as described in Section 2.4. The backward-facing step (BSF) simulation is then performed using these models to replace the traditional chemistry integration step. The surrogate system directly evolves seven representative scalars—temperature and six important species: O₂, OH, H₂O, CH₄, CO, and CO₂—predicted by the React-DeepONet model.

The remaining twelve non-representative species are recovered using the RecNet model, which maps the representative set to the full thermochemical state vector, as detailed in Section 2.2.3.

To ensure overall mass conservation, the mass fraction of nitrogen (N₂) is computed as:

$$Y_{N_2} = 1 - \sum_{i=1}^{18} Y_i \quad (3.1)$$

where the summation includes the six representative species and twelve reconstructed ones. Since nitrogen is inert, this treatment ensures conservation without altering the chemical kinetics.

To evaluate model accuracy, a baseline simulation using the standard CVODE-based chemistry integrator in PeleLMex is also conducted under identical flow and boundary conditions. The DeepONet-based results are then compared with this reference.

Figures 3.11 through 3.18 present side-by-side contours of temperature, methane, carbon monoxide, and hydroperoxyl mass fractions at different time instances, ranging from 0.5 ms to 2.75 ms. Each set includes the DeepONet prediction, the CVODE reference, and the corresponding absolute error field, defined as:

$$\text{Error} = \left| V_{\text{CVODE}} - V_{\text{React-DeepONet}} \right| \quad (3.2)$$

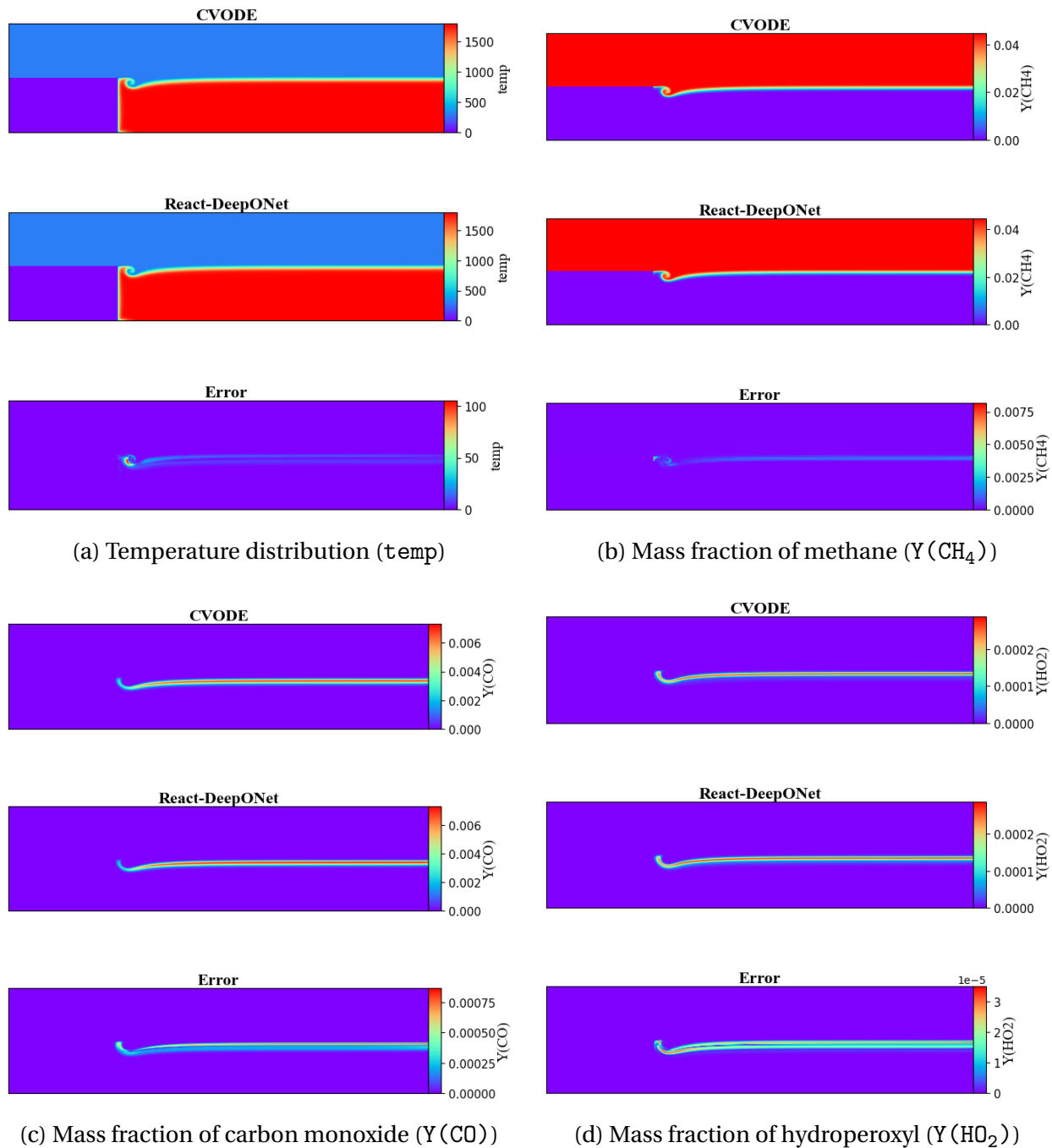
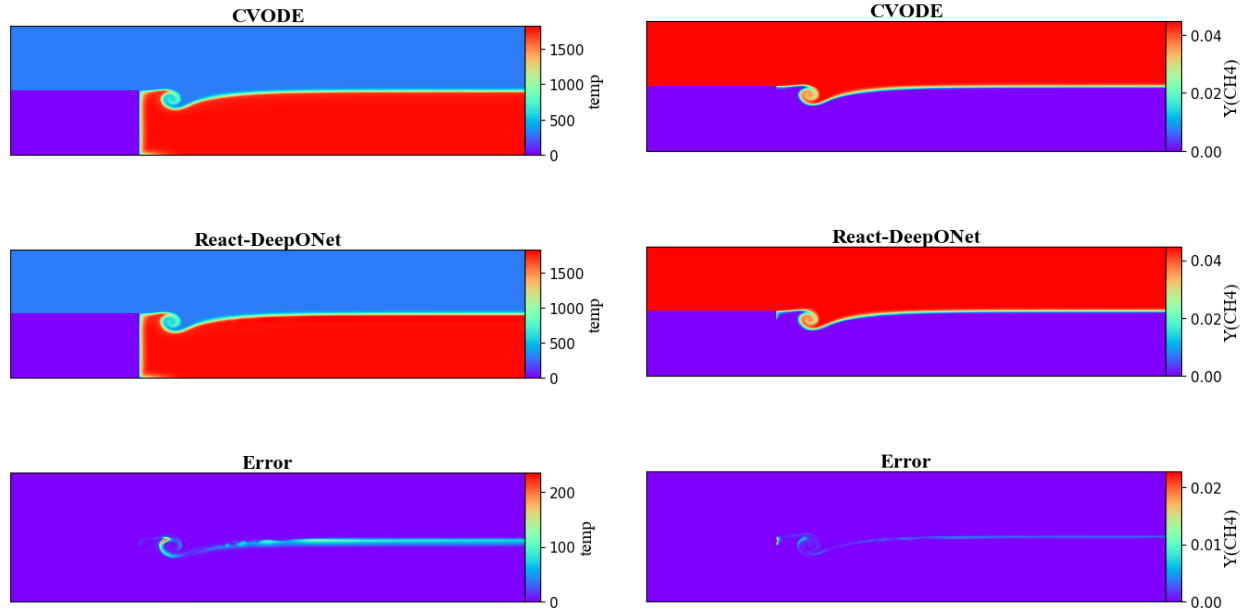


Figure 3.11: Comparison of CVODE and React-DeepONet predictions for key thermochemical variables at $t = 0.5$ ms in the Backward-Facing Step configuration. Each subplot shows results from detailed integration (CVODE), React-DeepONet predictions, and the corresponding error field.

At $t = 0.5$ ms, the predicted contours for temperature and species mass fractions ($Y(\text{CH}_4)$, $Y(\text{CO})$, and $Y(\text{HO}_2)$) from the React-DeepONet simulation are compared against those obtained from the CVODE-based detailed chemistry integration. As shown in Figures 3.11a–3.11d, React-DeepONet demonstrates high fidelity in reproducing the flame structure, capturing both the thermal field and species distribution with excellent spatial agreement. The temperature field shows accurate resolution of the flame front, with the location and thickness of the reaction zone closely matching CVODE results. The formation and propagation of the vortex near the backward-facing step is also well-represented, indicating that the auto-regressive DeepONet predictions are temporally and spatially coherent.

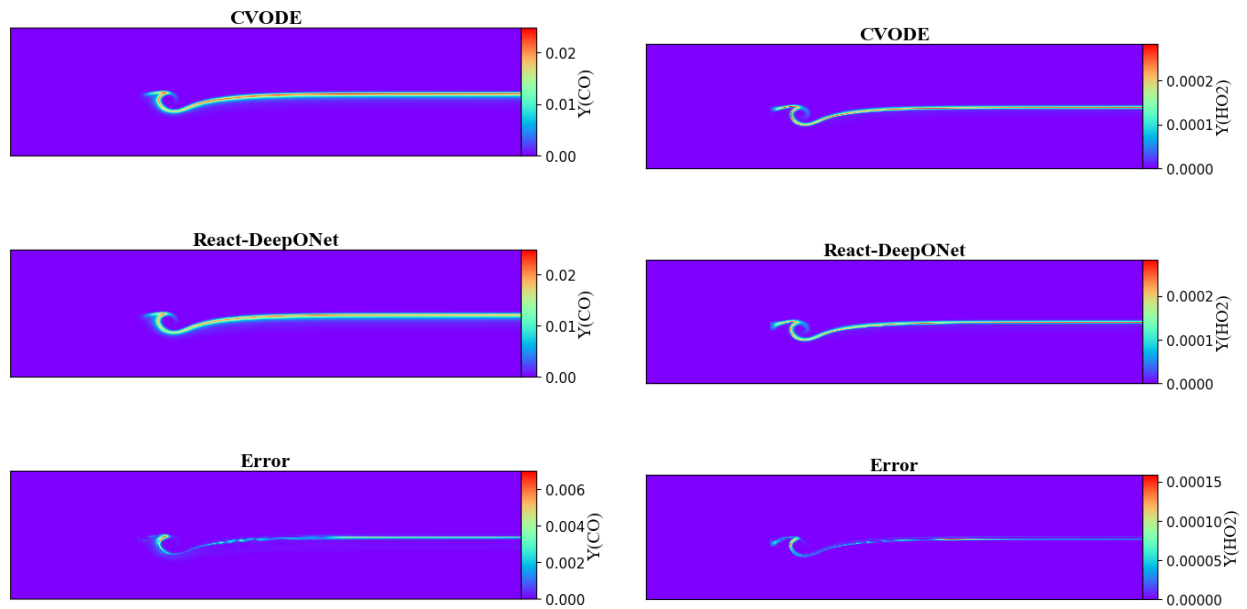
Among the plotted quantities, temperature, CH_4 , and CO are representative species directly predicted by React-DeepONet. Their strong agreement with CVODE establishes the validity of the model's learned dynamics within the core set of species used during training. On the other hand, HO_2 is a non-representative species, reconstructed from the representative scalars using the auxiliary RecNet. The accuracy observed in its spatial profile and peak concentrations confirms the effectiveness of RecNet in capturing the nonlinear correlations required for species reconstruction. The error contours in each subplot—computed as the absolute difference between CVODE and React-DeepONet results—show that the discrepancies are minimal and largely confined to thin regions around steep gradients or near vortex roll-ups. These observations validate that the React-DeepONet + RecNet surrogate framework preserves both reaction-driven and flow-driven features in a complex reacting flow, even at early times in the simulation.

At time snapshots $t = 1.0$ ms (Figure 3.12) and $t = 1.5$ ms (Figure 3.13), the Backward-Facing Step simulation exhibits the emergence and downstream advection of a prominent vortex structure originating at the corner of the step. This vortex results from the shear layer instability between the jet and the recirculating flow, and it plays a critical role in mixing unburnt reactants with combustion products, thereby influencing local flame propagation and stabilization.



(a) Temperature distribution (temp)

(b) Mass fraction of methane ($Y(\text{CH}_4)$)



(c) Mass fraction of carbon monoxide ($Y(\text{CO})$)

(d) Mass fraction of hydroperoxyl ($Y(\text{HO}_2)$)

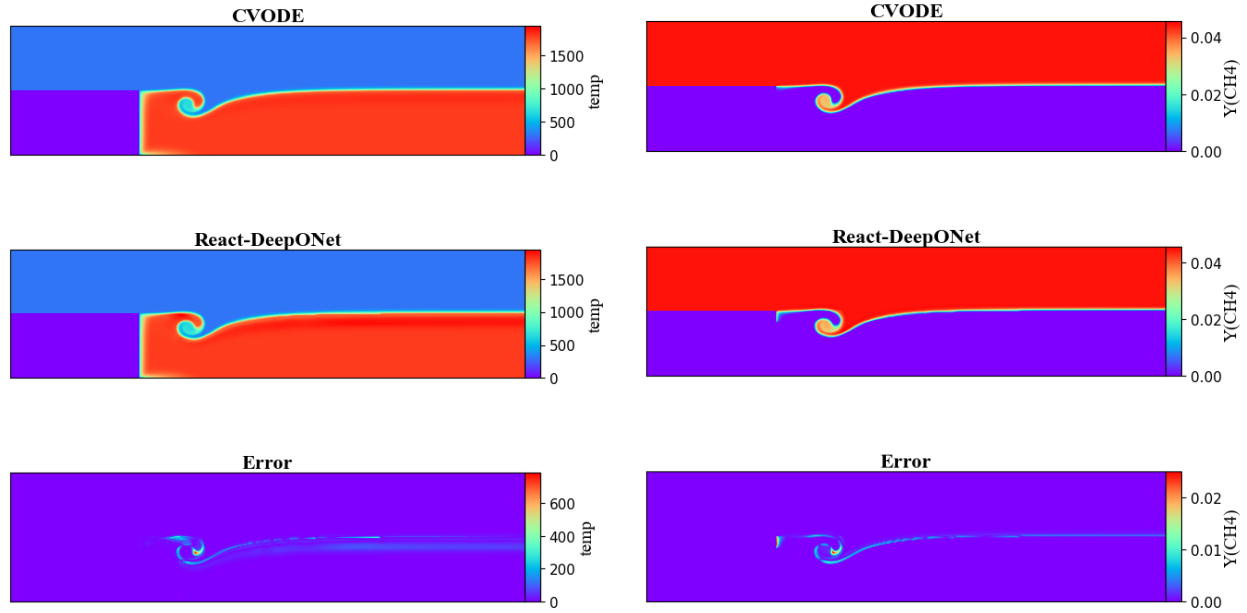
Figure 3.12: Comparison of CVODE and React-DeepONet predictions for key thermochemical variables at $t = 1$ ms in the Backward-Facing Step configuration. Each subplot shows results from detailed integration (CVODE), React-DeepONet predictions, and the corresponding error field.

The React-DeepONet model maintains high accuracy in predicting the representative species: temperature, CH_4 , and CO . In both figures, the temperature fields show that the model successfully captures the steep thermal gradients across the flame and the post-flame plateau. The CH_4 mass fraction fields accurately reflect the progressive consumption of methane, especially around the distorted flame front as it interacts with the vortex. CO predictions remain well-aligned with the CVODE baseline, with only slight underprediction near the core of the vortex and along the curved flame surface.

The HO_2 predictions—representing a non-representative species reconstructed via RecNet—show excellent agreement with the reference solution. Despite being indirectly inferred, the HO_2 field retains the sharp interface and post-flame gradients, reinforcing RecNet’s effectiveness in recovering the thermochemical state from a reduced set of representative inputs.

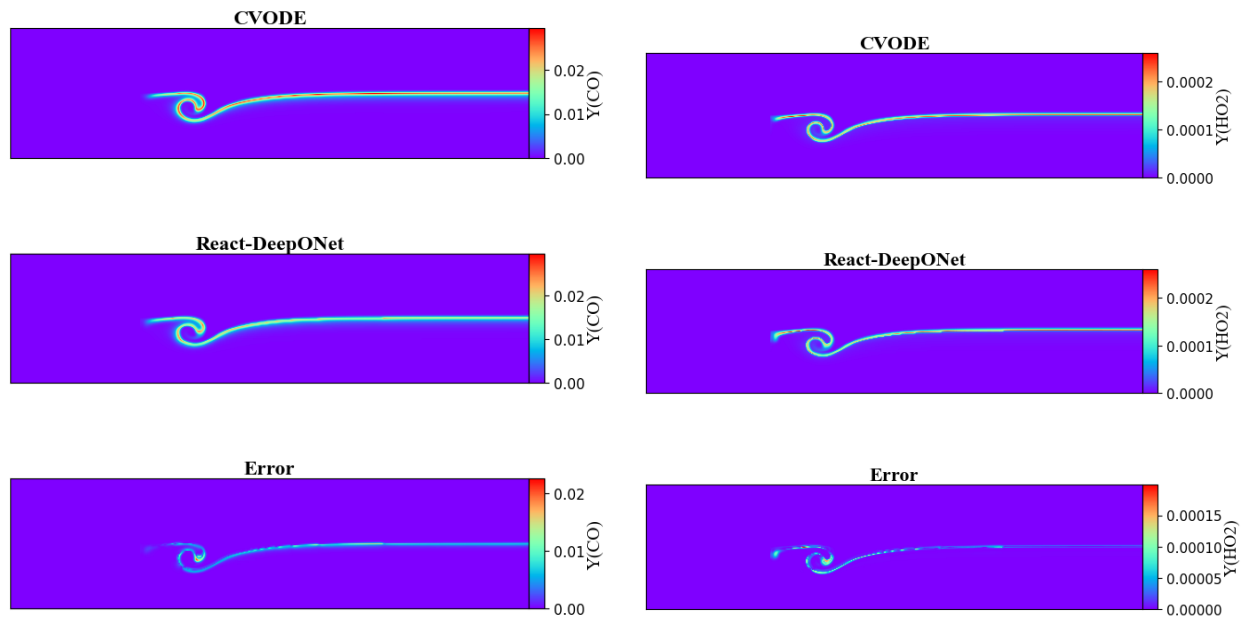
Between $t = 1.0$ ms and $t = 1.5$ ms, the increasing strength and size of the vortex introduces more pronounced curvature and localized strain along the flame front. This complexity leads to slightly elevated error magnitudes in the vicinity of the vortex, especially in temperature and CH_4 . Nevertheless, the errors remain spatially confined and do not compromise the global accuracy or stability of the surrogate model predictions. The fidelity observed across both representative and reconstructed variables strongly validates the applicability of the React-DeepONet–RecNet framework in capturing stiff chemistry dynamics under complex flow conditions.

At later times of 1.75 ms and 2.0 ms, shown in Figures 3.14 and 3.15, the flame structure becomes increasingly complex due to the roll-up of the vortex formed at the shear interface. Despite the unsteady nature of this flame front, the DeepONet model continues to show good agreement with the CVODE solution across all reported quantities. The agreement in temperature and representative species mass fractions such as Y_{CH_4} and Y_{CO} confirms the accurate behavior of the React-DeepONet model in reproducing the main combustion products and fuel consumption. Additionally, the reconstructed species Y_{HO_2} , predicted by the RecNet, aligns closely with the CVODE output, validating the reliability of the reconstruction process.



(a) Temperature distribution (temp)

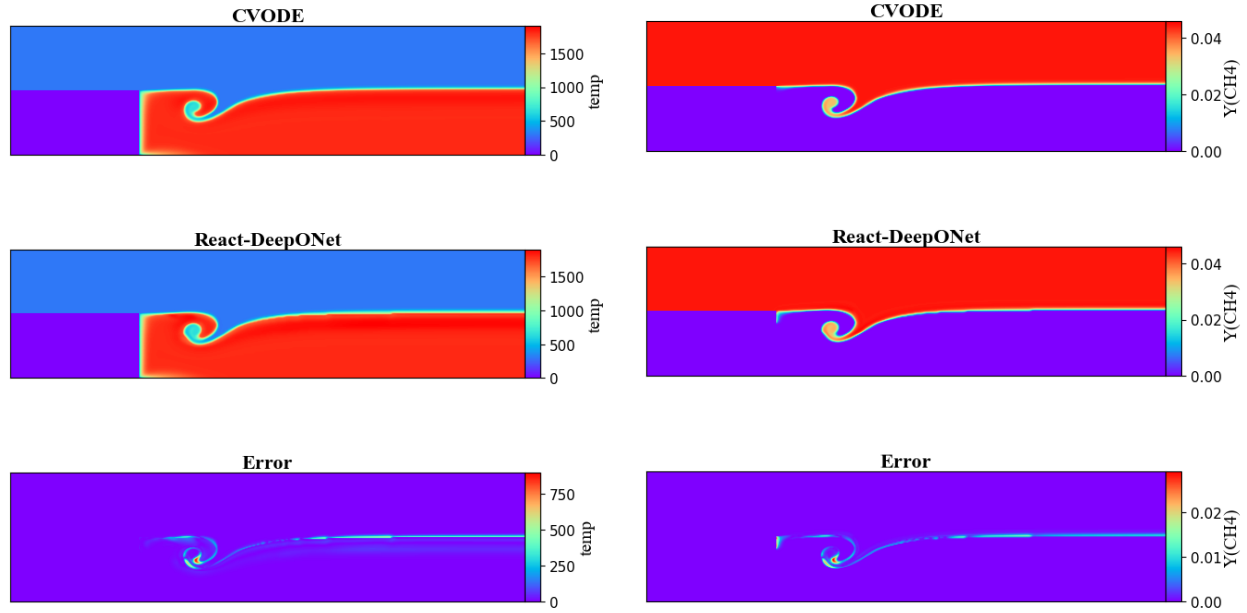
(b) Mass fraction of methane ($Y(\text{CH}_4)$)



(c) Mass fraction of carbon monoxide ($Y(\text{CO})$)

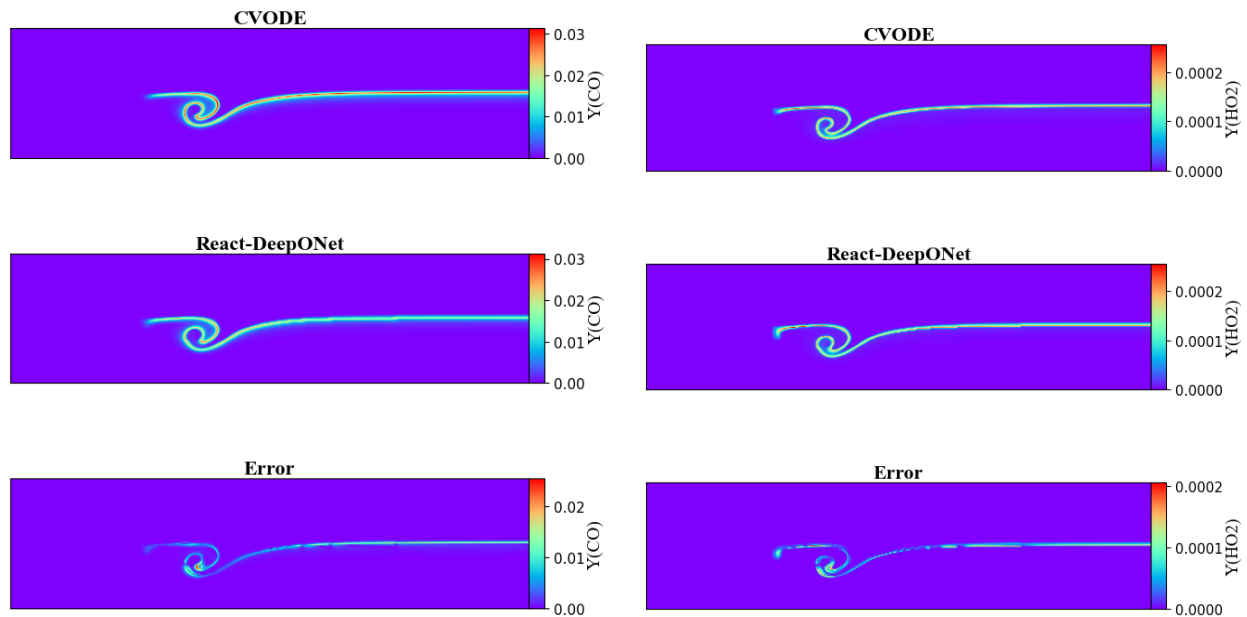
(d) Mass fraction of hydroperoxyl ($Y(\text{HO}_2)$)

Figure 3.13: Comparison of CVODE and React-DeepONet predictions for key thermochemical variables at $t = 1.5$ ms in the Backward-Facing Step configuration. Each subplot shows results from detailed integration (CVODE), React-DeepONet predictions, and the corresponding error field.



(a) Temperature distribution (temp)

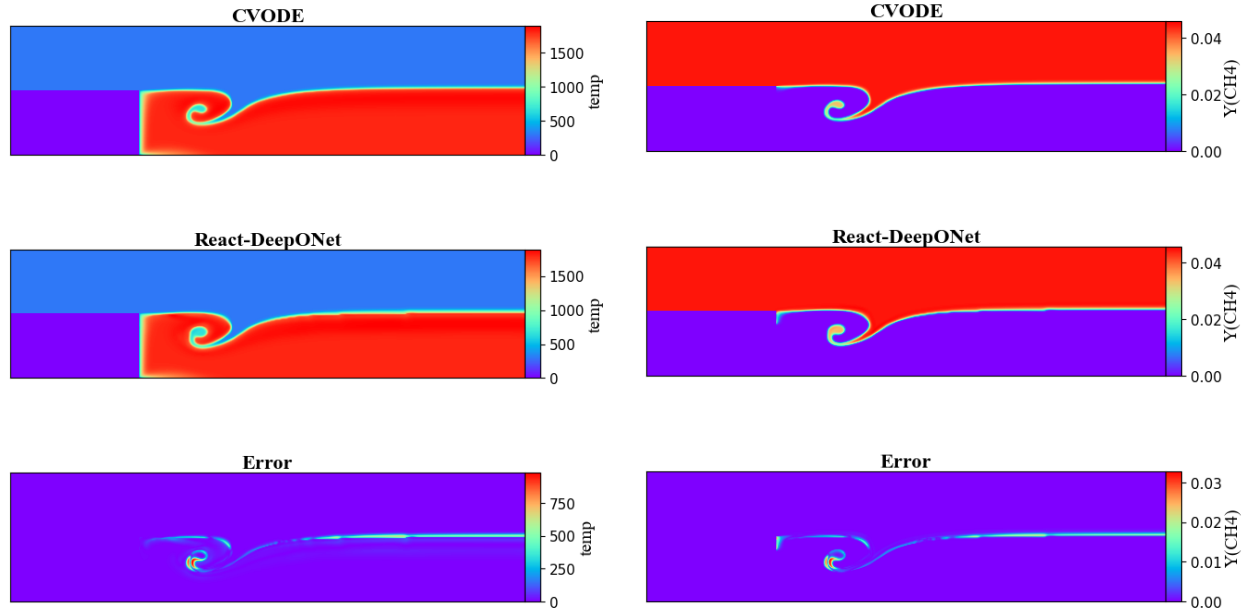
(b) Mass fraction of methane ($Y(\text{CH}_4)$)



(c) Mass fraction of carbon monoxide ($Y(\text{CO})$)

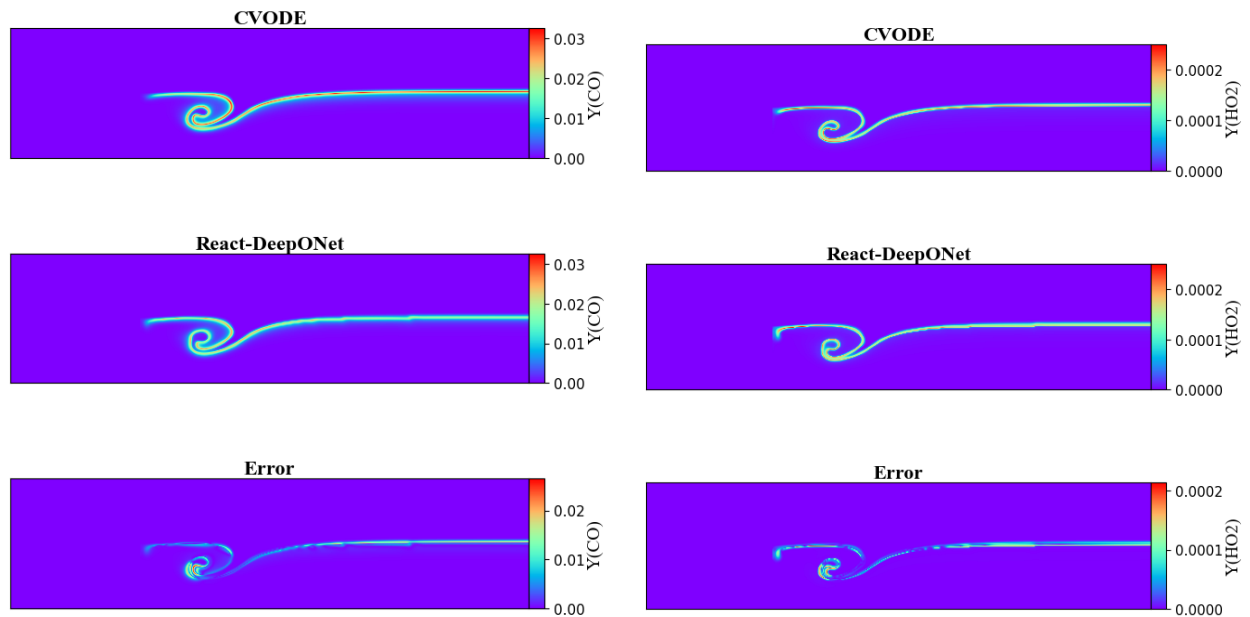
(d) Mass fraction of hydroperoxyl ($Y(\text{HO}_2)$)

Figure 3.14: Comparison of CVODE and React-DeepONet predictions for key thermochemical variables at $t = 1.75$ ms in the Backward-Facing Step configuration. Each subplot shows results from detailed integration (CVODE), React-DeepONet predictions, and the corresponding error field.



(a) Temperature distribution (temp)

(b) Mass fraction of methane ($Y(\text{CH}_4)$)



(c) Mass fraction of carbon monoxide ($Y(\text{CO})$)

(d) Mass fraction of hydroperoxyl ($Y(\text{HO}_2)$)

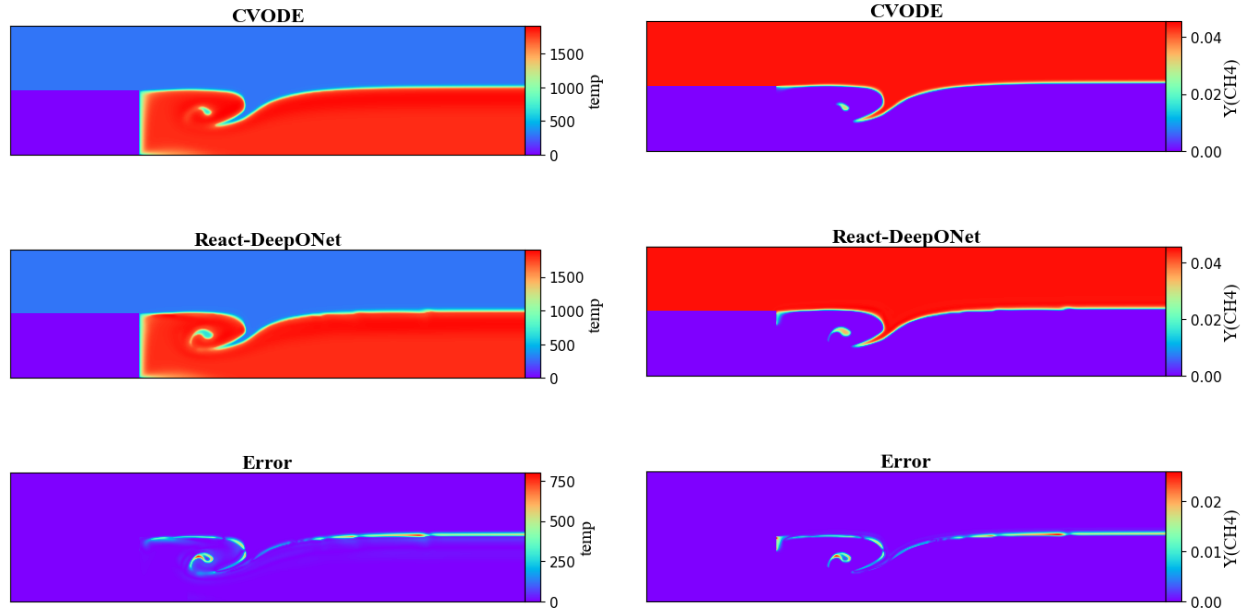
Figure 3.15: Comparison of CVODE and React-DeepONet predictions for key thermochemical variables at $t = 2$ ms in the Backward-Facing Step configuration. Each subplot shows results from detailed integration (CVODE), React-DeepONet predictions, and the corresponding error field.

However, as the vortex stretches and the scalar gradients become sharper, local discrepancies start to emerge, particularly in regions of high curvature and along the outer edges of the vortex core. This is reflected in the error contours, which show intensified values near the vortex ring at 2.0 ms. Yet, these errors remain physically consistent and localized, without distorting the global structure of the flame or its chemical composition trends. This further demonstrates the React-DeepONet’s robustness under complex, time-evolving combustion conditions.

At $t = 2.25$ ms and $t = 2.5$ ms, the flowfield undergoes a rapid morphological transformation, with the leading edge of the flame rolling onto itself and forming a localized recirculation structure. This vortex-like behavior introduces sharp gradients in temperature and species distributions, offering a challenging test for any surrogate model attempting to resolve stiff, nonlinear chemical kinetics in such dynamically evolving regions. As shown in Figs. 3.16 and 3.17, the React-DeepONet is able to closely reproduce the reference CVODE predictions across all four quantities: CH_4 , CO , HO_2 , and temperature.

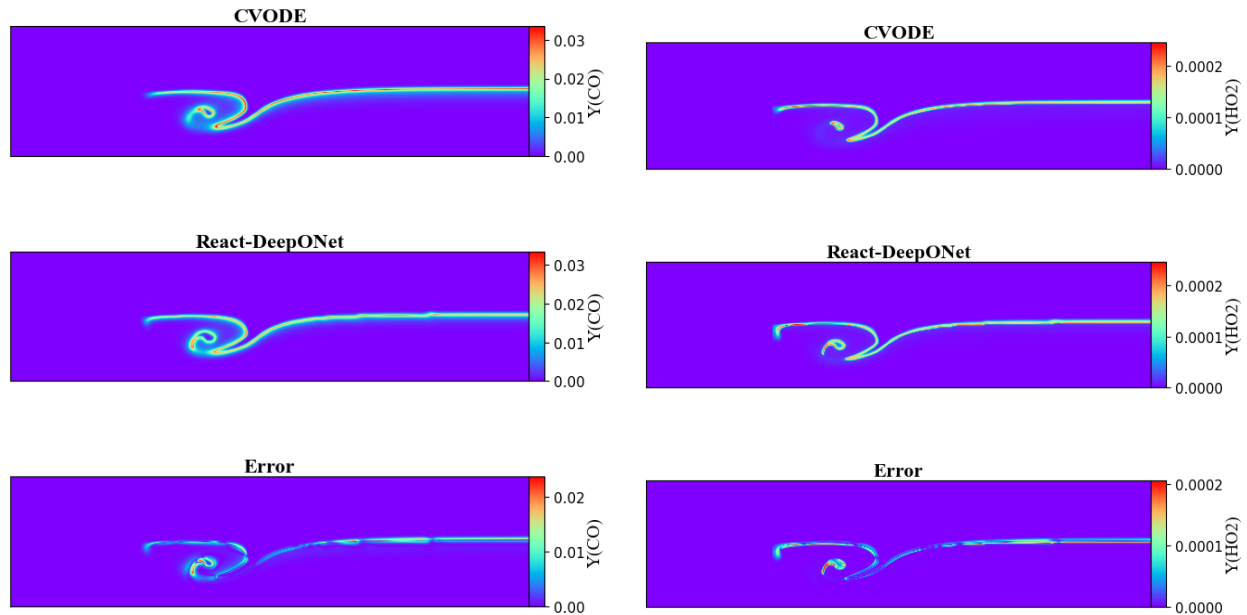
The flame structure is well captured in both magnitude and spatial location by the React-DeepONet model, including the crescent-shaped pocket of burned gases that forms due to the flame front interacting with the shear layer. While a slight delay in the DeepONet-predicted flame roll-up can be observed—most noticeably in CH_4 and temperature fields—this phase lag remains spatially confined and does not propagate downstream. Notably, even the indirectly inferred HO_2 field preserves the correct qualitative shape and localized high-gradient zones, reflecting the RecNet’s ability to reconstruct non-representative species from limited inputs with high fidelity.

Overall, despite the appearance of highly transient, vortex-dominated combustion features at these time instances, the React-DeepONet model continues to track the complex thermochemical evolution with minimal structural distortion or amplitude mismatch, highlighting its robustness in challenging reacting flow regimes.



(a) Temperature distribution (temp)

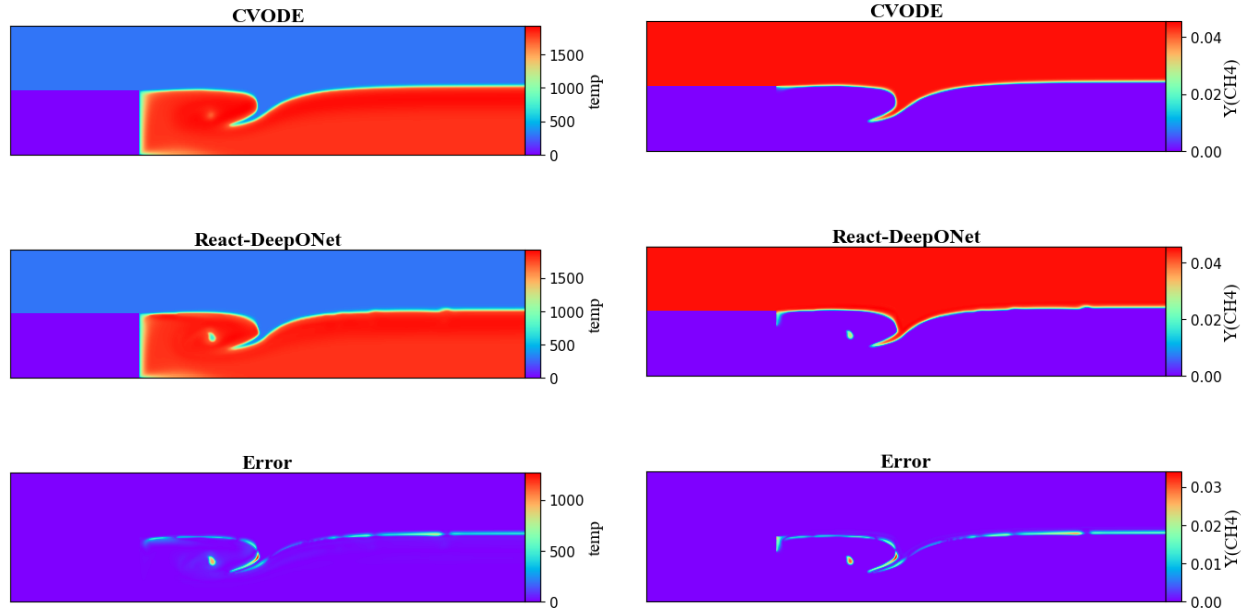
(b) Mass fraction of methane ($Y(\text{CH}_4)$)



(c) Mass fraction of carbon monoxide ($Y(\text{CO})$)

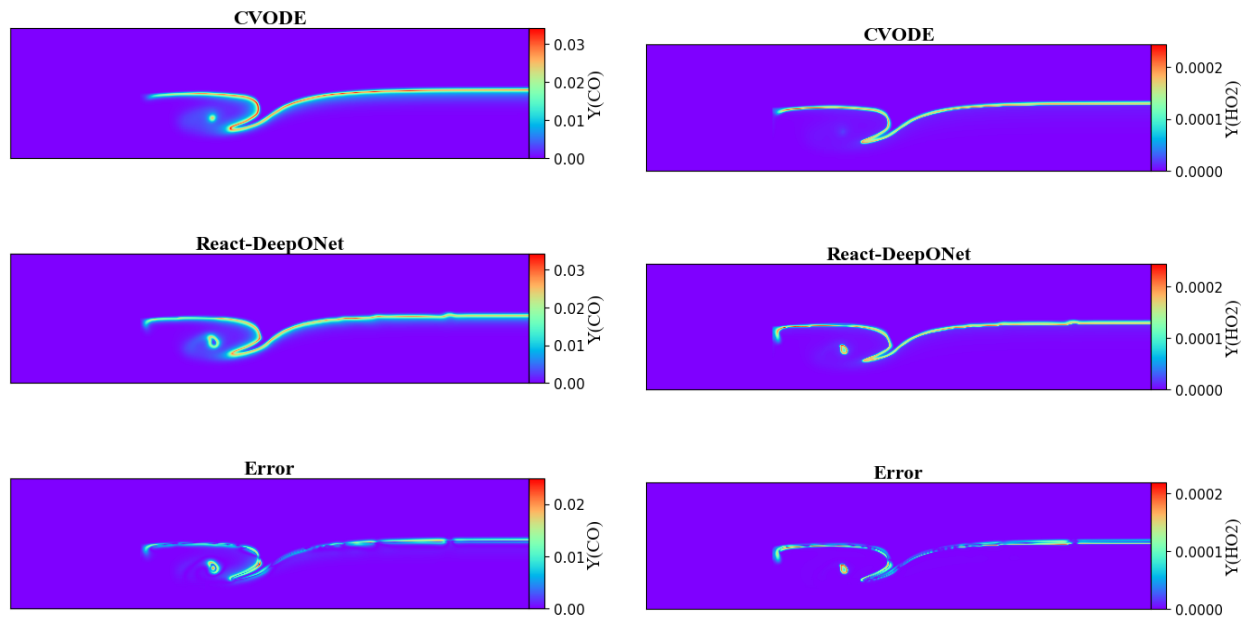
(d) Mass fraction of hydroperoxyl ($Y(\text{HO}_2)$)

Figure 3.16: Comparison of CVODE and React-DeepONet predictions for key thermochemical variables at $t = 2.25$ ms in the Backward-Facing Step configuration. Each subplot shows results from detailed integration (CVODE), React-DeepONet predictions, and the corresponding error field.



(a) Temperature distribution (temp)

(b) Mass fraction of methane ($Y(\text{CH}_4)$)



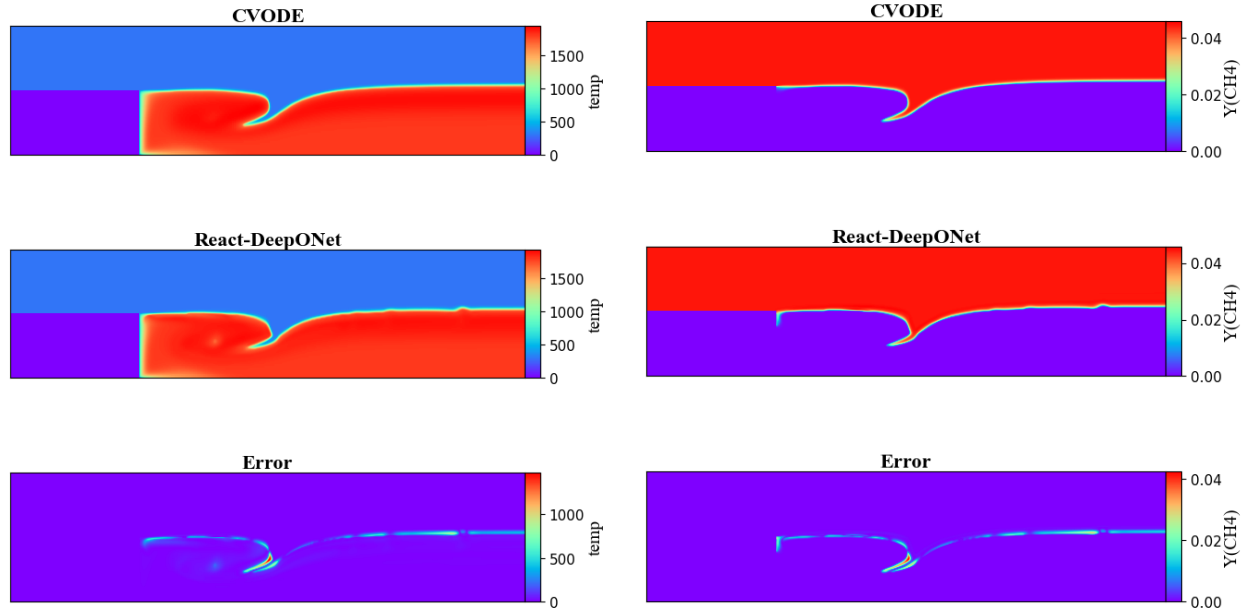
(c) Mass fraction of carbon monoxide ($Y(\text{CO})$)

(d) Mass fraction of hydroperoxyl ($Y(\text{HO}_2)$)

Figure 3.17: Comparison of CVODE and React-DeepONet predictions for key thermochemical variables at $t = 2.5$ ms in the Backward-Facing Step configuration. Each subplot shows results from detailed integration (CVODE), React-DeepONet predictions, and the corresponding error field.

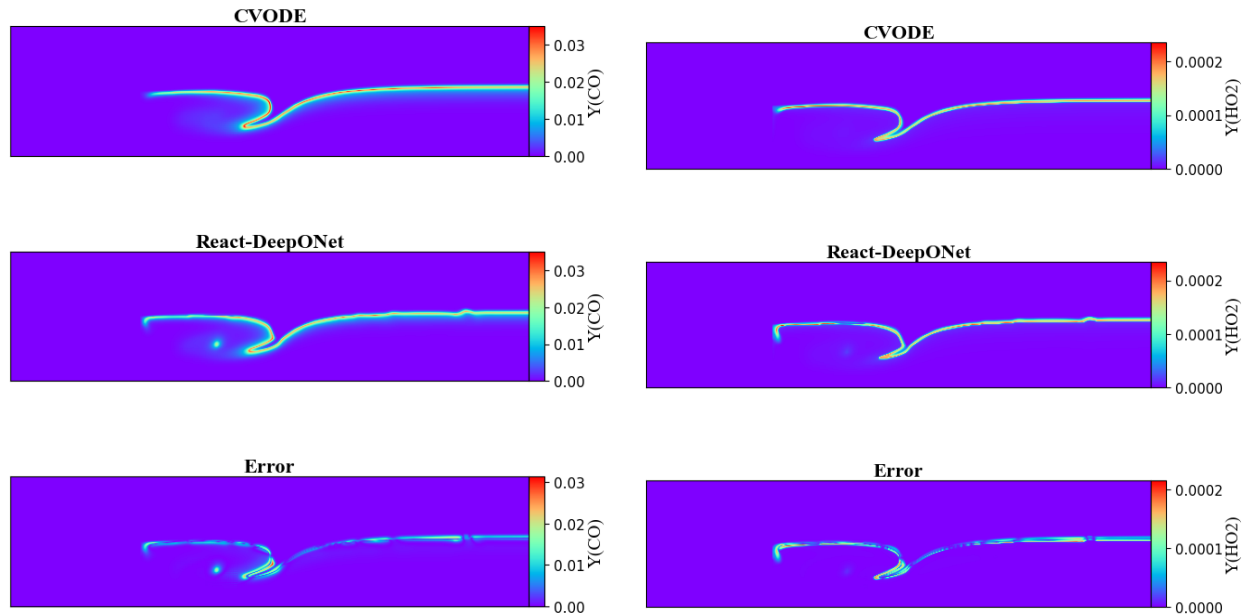
At $t = 2.75$ ms, the flame undergoes further topological changes as the previously formed roll-up structure elongates and convects downstream. The flame bubble begins to stretch and flatten along the shear layer, and complex gradients emerge particularly in the CO and HO₂ fields. Compared to the 2.5 ms frame, this evolution reflects increasing chemical and fluid dynamic complexity in the reaction zone.

React-DeepONet continues to predict the major flame features with high fidelity, reproducing the overall flame roll-up shape, scalar field gradients, and peak locations across all variables. However, a slight lag in the flame evolution remains visible relative to the CVODE reference, particularly near the leading edge of the CH₄ front and in the peak CO and HO₂ contours. This lag introduces localized increases in error, most noticeably in the core of the recirculation bubble and along the reaction front. Despite this, the model maintains excellent agreement across the domain, underscoring its robustness even under topologically evolving and chemically stiff scenarios.



(a) Temperature distribution (temp)

(b) Mass fraction of methane ($Y(\text{CH}_4)$)



(c) Mass fraction of carbon monoxide ($Y(\text{CO})$)

(d) Mass fraction of hydroperoxyl ($Y(\text{HO}_2)$)

Figure 3.18: Comparison of CVODE and React-DeepONet predictions for key thermochemical variables at $t = 2.75$ ms in the Backward-Facing Step configuration. Each subplot shows results from detailed integration (CVODE), React-DeepONet predictions, and the corresponding error field.

3.3.2 Computational Performance and Speed-Up

To evaluate the computational efficiency achieved by deploying surrogate models for chemistry integration, we compare the total simulation cost of the backward-facing step (BSF) case between two setups: one using the default CVODE-based chemical integrator in PeleLMEx, and the other using the trained React-DeepONet and RecNet models. Both simulations were run under identical flow and boundary conditions for a final physical time of 3 milliseconds.

In reacting flow simulations, the overall computational cost is typically dominated by the combustion component—that is, the cost of integrating the stiff chemical kinetics. Other components such as continuity, momentum, pressure, energy, species transport, and turbulence contribute to the transport cost. In the case of the CVODE-based simulation, this transport cost is negligible compared to the combustion cost. However, in the React-DeepONet-based simulation, the combustion and transport costs become comparable, highlighting the substantial reduction in chemistry integration expense achieved through surrogate modeling.

For this configuration, the DRM19 mechanism was used, with six representative species predicted using the React-DeepONet and twelve non-representative species reconstructed using the RecNet. This setup led to an observed speedup of approximately 11% in the total wall-clock time for the simulation.

Although this speedup is meaningful, it remains moderate. The reason lies in the relatively small size and simplicity of the DRM19 mechanism; the cost of CVODE-based integration for such a mechanism is not prohibitively high. However, this scenario still illustrates the practical viability of integrating DeepONet-based surrogates into a high-fidelity CFD solver. In future applications involving more complex fuels—such as n-dodecane or jet fuel surrogates—the cost of ODE-based chemical integration can rise dramatically due to the increased number of species and reactions. In such cases, the expected speedup from using surrogate models like React-DeepONet is significantly higher.

It is worth noting that the total cost of surrogate-based simulations includes not only the model prediction time but also the overhead from Python-C data conversion and communica-

tion, as React-DeepONet is implemented in Python and called from the C++-based PeleLMeX framework. Despite this, the use of JAX and its Just-In-Time (JIT) compilation feature for efficient model inference, along with optimized NumPy-C API calls for seamless memory sharing, contributes significantly to the observed speedup. These design choices in the surrogate integration pipeline reduce the per-cell prediction latency and demonstrate the feasibility of coupling modern ML frameworks with traditional high-performance CFD solvers.

3.4 Conclusion

In this work, we present an acceleration framework for integrating complex fuel chemistry in computational fluid dynamics (CFD) simulations using a data-driven surrogate modeling approach. At the heart of this framework lies the React-DeepONet, a deep neural operator that directly predicts the evolution of a subset of key thermochemical scalars—referred to as representative scalars—over adaptive time steps. This surrogate model replaces the traditional stiff ODE-based solver (CVODE) within the chemistry integration step of the PeleLMeX solver.

To recover the complete thermochemical state required for the simulation, a reconstruction network (RecNet) is deployed alongside React-DeepONet. RecNet accurately infers the remaining non-representative species using the predicted representative scalars, enabling consistent species profiles and preserving fidelity in the reactive flow modeling.

React-DeepONet and RecNet are integrated into the PeleLMeX solver by replacing the pre-existing chemistry integrator. Efficient coupling between the JAX-based Python inference engine and the C++ codebase of PeleLMeX is achieved through the Python-C API, with just-in-time compilation (JAX JIT) significantly reducing model inference cost, and shared memory buffers via NumPy-C API minimizing data transfer latency.

The integrated surrogate framework is evaluated using the canonical backward-facing step flame configuration. The results demonstrate excellent agreement between the React-DeepONet predictions and the CVODE-based detailed simulation. The representative scalars,

including temperature, methane, and carbon monoxide, are accurately captured, while the reconstructed scalar (e.g., HO₂) also exhibits strong agreement with the reference solution. The framework captures transient phenomena such as flame folding, vortex formation, and scalar gradient evolution with high accuracy.

A total simulation speedup of approximately 11% is achieved for the DRM19 mechanism. While this gain is moderate due to the relatively small size of the DRM19 mechanism, the result illustrates the potential of the framework. For larger and more complex reaction mechanisms, such as those involving *n*-dodecane, where traditional solvers incur significantly higher computational cost, the proposed surrogate strategy is expected to offer substantial performance benefits.

In summary, the React-DeepONet and RecNet integration into PeleLMex demonstrates a promising advancement in accelerating detailed chemistry in CFD through machine learning. This approach maintains high predictive accuracy while offering computational efficiency, marking a step forward in the deployment of neural surrogate models for practical reactive flow simulations.

CHAPTER

4

CONCLUSIONS AND FUTURE WORKS

4.1 Summary

This dissertation presents a machine learning-based framework to accelerate chemical kinetics integration in low Mach number CFD solvers. The key idea is to replace the traditional, computationally intensive ODE-based chemistry integration routine with a surrogate model—React-DeepONet—that learns to evolve a subset of the thermochemical state vector over adaptive time steps. The framework is developed, integrated, and validated within the context of the PeleLMeX solver using a canonical reactive flow problem.

Chapter 1 introduces the motivation for this work, with a focus on the challenges posed by reaction integration in reactive flow solvers. Operator splitting methods—while effective—still rely on stiff ODE solvers like CVODE, which dominate the overall simulation cost, particularly

for detailed reaction mechanisms.

Chapter 2 details the methodology. The chapter begins with an overview of PeleLMeX, including its modular structure and the strategy used for integrating chemical reactions via CVODE. This is followed by an introduction to the React-DeepONet architecture and the reconstruction network (RecNet), which together model the full thermochemical state evolution. The chapter also discusses how representative scalars are selected and how RecNet recovers the non-representative species. Data generation strategies using Cantera, including 0D evolutions from sampled states, are briefly outlined. The integration challenges posed by embedding Python-based ML inference into a C++-MPI-GPU solver are addressed, and the modifications made to reactor classes, memory management, and data transfer routines are summarized.

Chapter 3 demonstrates the application of the proposed approach to a well-known test case: a backward-facing step (BFS) laminar premixed flame. The setup, boundary conditions, and initial conditions are described in detail. Representative data from the BFS simulation is sampled and evolved in Cantera to produce training trajectories for React-DeepONet. After training, the model is validated offline on 0D trajectories and then integrated into PeleLMeX. Simulation results from the React-DeepONet-integrated solver are compared against the baseline CVODE-based simulation. Contour plots for representative and reconstructed scalars, along with absolute error fields, confirm the surrogate model's ability to capture flame dynamics, vortex structures, and transient species evolution with high fidelity. Finally, a simulation speedup of 11% is achieved using the DRM19 mechanism, showcasing the promise of this method. While moderate for this relatively small mechanism, the approach is expected to scale efficiently for larger kinetic models.

Overall, this work highlights the potential of deep operator networks in accelerating high-fidelity combustion simulations while preserving predictive accuracy—laying the groundwork for future integration into complex and realistic reactive flow configurations.

4.2 Future Work

This dissertation lays the groundwork for accelerating chemical kinetics in CFD solvers using a surrogate modeling framework based on React-DeepONet and RecNet. While the current implementation has demonstrated promising results in a two-dimensional backward-facing step flame configuration, several avenues remain open for further development and extension.

A key direction is to apply the integrated framework to more realistic and computationally intensive problems. Future work should focus on deploying React-DeepONet within fully three-dimensional turbulent combustion simulations involving complex fuels such as *n*-dodecane. Given the significantly larger and stiffer chemical mechanisms associated with such fuels, it is anticipated that the computational savings achieved by bypassing traditional ODE-based reaction solvers will be much more substantial than those observed in the current study.

Another area of improvement lies in the data generation strategy used to train the machine learning models. The present approach relies on sampling from CFD simulations, which inherently ties the model training to a specific setup. Exploring alternative strategies such as generating data from large batches of zero-dimensional constant-volume reactors or perfectly stirred reactors (PSRs) could provide a more comprehensive thermochemical coverage. This would allow the models to learn broader reaction behavior without the need for prior CFD runs, ultimately improving training efficiency and model generality.

Lastly, future work should aim to develop generalized React-DeepONet models for a given fuel that can be applied across different CFD configurations without retraining. Achieving such generalization would enhance the versatility and reusability of the framework, making it more attractive for deployment in a wide range of practical combustion applications. This direction would involve investigating training methods and network architectures capable of capturing the intrinsic behavior of the fuel across varying flow and thermodynamic conditions.

REFERENCES

- [1] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via deepoNet based on the universal approximation theorem of operators," *Nature machine intelligence*, vol. 3, no. 3, pp. 218–229, 2021.
- [2] A. Kumar and T. Echehki, "Combustion chemistry acceleration with deepoNets," *Fuel*, vol. 365, p. 131212, 2024.
- [3] J. A. Badra, F. Khaled, M. Tang, Y. Pei, J. Kodavasal, P. Pal, O. Owoyele, C. Fuetterer, B. Matia, and F. Aamir, "Engine combustion system optimization using computational fluid dynamics and machine learning: a methodological approach," *Journal of energy resources technology*, vol. 143, no. 2, p. 022306, 2021.
- [4] V. Vilag, J. Vilag, R. Carlanescu, A. Mangra, and F. Florean, "Cfd application for gas turbine combustion simulations," in *Computational Fluid Dynamics Simulations*, IntechOpen, 2019.
- [5] J. H. Chen, "Petascale direct numerical simulation of turbulent combustion—fundamental insights towards predictive models," *Proceedings of the Combustion Institute*, vol. 33, no. 1, pp. 99–123, 2011.
- [6] Y. Gao, Y. Liu, Z. Ren, and T. Lu, "A dynamic adaptive method for hybrid integration of stiff chemistry," *Combustion and Flame*, vol. 162, no. 2, pp. 287–295, 2015.
- [7] J. H. Chen, A. Choudhary, B. De Supinski, M. DeVries, E. R. Hawkes, S. Klasky, W.-K. Liao, K.-L. Ma, J. Mellor-Crummey, N. Podhorszki, *et al.*, "Terascale direct numerical simulations of turbulent combustion using s3d," *Computational Science & Discovery*, vol. 2, no. 1, p. 015001, 2009.
- [8] L. Zhou, Y. Song, W. Ji, and H. Wei, "Machine learning for combustion," *Energy and AI*, vol. 7, p. 100128, 2022.
- [9] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.
- [10] V. Raman and M. Hassanaly, "Emerging trends in numerical simulations of combustion systems," *Proceedings of the Combustion Institute*, vol. 37, no. 2, pp. 2073–2089, 2019.
- [11] S. A. Kalogirou, "Artificial intelligence for the modeling and control of combustion processes: a review," *Progress in energy and combustion science*, vol. 29, no. 6, pp. 515–566, 2003.
- [12] D. J. Gardner, D. R. Reynolds, C. S. Woodward, and C. J. Balos, "Enabling new flexibility in the SUNDIALS suite of nonlinear and differential/algebraic equation solvers," *ACM Transactions on Mathematical Software (TOMS)*, vol. 48, no. 3, pp. 1–24, 2022.

- [13] S. B. Pope, “Computationally efficient implementation of combustion chemistry using in situ adaptive tabulation,” 1997.
- [14] S. R. Tonse, N. W. Moriarty, M. Frenklach, and N. J. Brown, “Computational economy improvements in prism,” *International Journal of Chemical Kinetics*, vol. 35, no. 9, pp. 438–452, 2003.
- [15] U. Maas and S. B. Pope, “Implementation of simplified chemical kinetics based on intrinsic low-dimensional manifolds,” in *Symposium (international) on combustion*, vol. 24, pp. 103–112, Elsevier, 1992.
- [16] S. Lam and D. Goussis, “The csp method for simplifying kinetics,” *International journal of chemical kinetics*, vol. 26, no. 4, pp. 461–486, 1994.
- [17] J. C. Sutherland and A. Parente, “Combustion modeling using principal component analysis,” *Proceedings of the Combustion Institute*, vol. 32, no. 1, pp. 1563–1570, 2009.
- [18] O. Owoyele and T. Echekeki, “Toward computationally efficient combustion dns with complex fuels via principal component transport,” *Combustion Theory and Modelling*, vol. 21, no. 4, pp. 770–798, 2017.
- [19] M. R. Malik, P. O. Vega, A. Coussement, and A. Parente, “Combustion modeling using principal component analysis: A posteriori validation on sandia flames d, e and f,” *Proceedings of the Combustion Institute*, vol. 38, no. 2, pp. 2635–2643, 2021.
- [20] W. Ji and S. Deng, “Autonomous discovery of unknown reaction pathways from data by chemical reaction neural network,” *The Journal of Physical Chemistry A*, vol. 125, no. 4, pp. 1082–1092, 2021.
- [21] T. Zhang, Y. Zhang, Y. Ju, *et al.*, “A deep learning-based ode solver for chemical kinetics,” *arXiv preprint arXiv:2012.12654*, 2020.
- [22] O. Owoyele and P. Pal, “Chemnode: A neural ordinary differential equations framework for efficient chemical kinetic solvers,” *Energy and AI*, vol. 7, p. 100118, 2022.
- [23] S. Kim, W. Ji, S. Deng, Y. Ma, and C. Rackauckas, “Stiff neural ordinary differential equations,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 9, 2021.
- [24] C. J. Balos, M. Day, L. Esclapez, A. M. Felden, D. J. Gardner, M. Hassanaly, D. R. Reynolds, J. S. Rood, J. M. Sexton, N. T. Wimer, *et al.*, “Sundials time integrators for exascale applications with many independent systems of ordinary differential equations,” *The International Journal of High Performance Computing Applications*, vol. 39, no. 1, pp. 123–146, 2025.
- [25] L. Esclapez, M. Day, J. Bell, A. Felden, C. Gilet, R. Grout, M. Henry de Frahan, E. Motheau, A. Nonaka, L. Owen, B. Perry, J. Rood, N. Wimer, and W. Zhang, “PeleLMEx: an AMR Low Mach Number Reactive Flow Simulation Code without level sub-cycling,” *Journal of Open Source Software*, vol. 8, no. 90, p. 5450, 2023.

- [26] W. Zhang, A. Almgren, V. Beckner, J. Bell, J. Blaschke, C. Chan, M. Day, B. Friesen, K. Gott, D. Graves, M. Katz, A. Myers, T. Nguyen, A. Nonaka, M. Rosso, S. Williams, and M. Zingale, "AMReX: a framework for block-structured adaptive mesh refinement," *Journal of Open Source Software*, vol. 4, p. 1370, May 2019.
- [27] M. T. Henry de Frahan, L. Esclapez, J. Rood, N. T. Wimer, P. Mullaney, B. A. Perry, L. Owen, H. Sitaraman, S. Yellapantula, M. Hassanaly, M. J. Rahimi, M. J. Martin, O. A. Doronina, S. N. A., M. Rieth, W. Ge, R. Sankaran, A. S. Almgren, W. Zhang, J. B. Bell, R. Grout, M. S. Day, and J. H. Chen, "The pele simulation suite for reacting flows at exascale," *Proceedings of the 2024 SIAM Conference on Parallel Processing for Scientific Computing*, pp. 13–25, 2024.
- [28] A. Nonaka, M. S. Day, and J. B. Bell, "A conservative, thermodynamically consistent numerical approach for low mach number combustion. part i: Single-level integration," *Combustion Theory and Modelling*, vol. 22, no. 1, pp. 156–184, 2018.
- [29] K. M. Gitushi, R. Ranade, and T. Echekeki, "Investigation of deep learning methods for efficient high-fidelity simulations in turbulent combustion," *Combustion and Flame*, vol. 236, p. 111814, 2022.
- [30] S. Goswami, A. D. Jagtap, H. Babaei, B. T. Susi, and G. E. Karniadakis, "Learning stiff chemical kinetics using extended deep neural operators," *Computer Methods in Applied Mechanics and Engineering*, vol. 419, p. 116674, 2024.
- [31] Z. Mao, L. Lu, O. Marxen, T. A. Zaki, and G. E. Karniadakis, "Deepm&mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators," *Journal of computational physics*, vol. 447, p. 110698, 2021.
- [32] S. Wang, H. Wang, and P. Perdikaris, "Learning the solution operator of parametric partial differential equations with physics-informed deep neural networks," *Science advances*, vol. 7, no. 40, p. eabi8605, 2021.
- [33] A. Kumar and T. Echekeki, "A framework for combustion chemistry acceleration with deep neural networks," *arXiv preprint arXiv:2304.12188*, 2023.
- [34] S. Alqahtani and T. Echekeki, "A data-based hybrid model for complex fuel chemistry acceleration at high temperatures," *Combustion and Flame*, vol. 223, pp. 142–152, 2021.
- [35] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.
- [36] D. G. Goodwin, R. L. Speth, H. K. Moffat, and B. W. Weber, "Cantera: An object-oriented software toolkit for chemical kinetics, thermodynamics, and transport processes," *Zenodo*, 2018.