Reduced-Complexity Modeling of Multi-Scale Problems in the Low Data Regime

by

Jiayang Xu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in The University of Michigan
2022

Doctoral Committee:

Associate Professor Karthik Duraisamy, Chair
Professor Krzysztof Fidkowski
Assistant Professor Cheng Huang
Associate Professor Eric Johnsen

Jiayang Xu

davidxu@umich.edu

ORCID iD: 0000-0002-9567-124X

For my parents.

ACKNOWLEDGEMENTS

Thank you for your time and interest if you are reading this part. I would like to take the privilege to acknowledge the great people I met during my journey in Ann Arbor. Before anyone else is my advisor, Professor Karthik Duraisamy. I first met him in his welcome talk at the department orientation for new graduate students. Honestly, limited by my English listening and background knowledge at the time, I could not follow all of the details. However, I was deeply excited by his passionate speech, and the blueprint he depicted about studying at the University of Michigan and conducting research about scientific computing. I was fortunate enough to be his student in CFD-I, and to join his lab afterward. Over the past 6 years, in classrooms and offices, I asked countless questions, mostly starting with "I have a stupid question", and he always replies with "there is no stupid question". He granted me enough freedom in research, and at the same time guided me on how to formulate, study, conclude, and present every single problem rigorously before pivoting. He devoted a tremendous amount of time discussing and advising every single piece of work of mine. This paragraph should have been summarized with words like "wisdom" and "vision", but they are not special enough to describe my incredible advisor. I am very grateful to him.

Secondly, I would like to thank my other committee members, Professors Krzysztof Fidkowski, Eric Johnsen, and Cheng Huang, for their support and comments since the planning stage of the thesis. I want to give additional thanks to Prof. Fidkowski for the knowledge I gained from his computational methods course. I believe he provides

a lifetime warranty for his courses since he never rejected my bothers with various questions long after graduating from the class. I also want to give special thanks to Cheng. He joined our lab as a research scientist a few years ago, since when I had the privilege to work with him on the same research project. He helped everyone in the project by educating us about combustion simulation, organizing meetings, and designing test cases that shapes the whole project. He is also a good friend outside the lab, leading me during my first conferences, and also showing me around Ann Arbor, although arrived later than I did.

I also enjoyed tremendous support and help from everyone in CASLAB. Dr. Davoudi encouraged me to continue into the Ph.D. program together with him when I was not self-confident. I learned a lot from his spirit of challenging, experiencing, and climbing. Chris and Nick deserve special thanks for taking my daily bugs patiently. I never learned how to respond properly when they sometimes say "sorry I couldn't be of more help". Hangouts with Hindi speakers, Aniruddhe, Vishal, and Shaowu were great refreshing moments during busy work. Our "employee benefits" also kept improving over the years. Official group meeting food has been switched from pizzas to sandwiches for our health. Recently, a coffee machine also arrived. It is a two-in-one model supporting both ground coffee and capsules, which made me very proud in front of students from other labs. The happy moments shared by everyone in the weekly online social meetings during the pandemic were relaxing and healing.

The journey could not have been completed without the companion of my friends. In particular, I would like to thank my roommate Zhe "Dashen" Du. We came to the U.S. together and shared all the happy moments. I took the initial challenges in a new environment with my friends in Aerospace Engineering, Guodong Chen, Xunwei "Jack" Zhang, Yucheng Liu, Di Wu, Pengxin Zhao, and Chenxing Yu. Over the years, I had several wonderful trips across the country with Shaowu Pan, Xingzuo & Xingyou Wang, Zezhi Zhang, Chenlan Wang & Chen Li, and Di "Wendy" Zhang.

Special thanks should be given to my warriors in the King's Valley, Bruce Huang, Ziyi Ye, and Tingting Liu; we are the best team! In addition, Tingting is an excellent psychologist who supported me largely during the pandemic. I am also fortunate to receive remote supports from friends in other cities of the planet, especially from Zhe Yan, Shusen "Sensen" Zhang, Junchen "Dahuang" Huang, Han Gao, and Tianjiao Zhou.

Lastly and most importantly, I want to thank my parents. Since my childhood, they have kept saying they could not help or guide me on this or that, and have encouraged me to make decisions as I wish. I enjoy the freedom, and at the same time, I am sure I would have achieved nothing without their support and encouragement. I also wish to mention our family dog, Xiaohuo, who brought happiness to our family, especially to my parents when I am abroad. He always has a special position in our memory.

The research in this thesis is supported by the Air Force under the Center of Excellence grant FA9550-17-1-0195, titled *Multi-Fidelity Modeling of Rocket Combustor Dynamics* (Program Managers: Dr. Mitat Birken and Dr. Fariba Fahroo).

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

LIST OF ABBREVIATIONS

BC boundary condition

BDF backward difference formula

CAE convolutional autoencoder

CFD computational fluid dynamics

CNN convolutional neural network

CP conditional parameterized/parameterization

CP-GNet conditional parameterized graph neural network

CPS conventional parallel staggered

CSS conventional serial staggered

CVRC continuously variable resonance combustor

DEIM discrete empirical interpolation method

DG discontinuous Galerkin

DL deep learning

DNS direct numerical simulation

DoF degree of freedom

FD-FD full-domain training and full-domain prediction

FD-MD full-domain training and multi-domain prediction

FDM finite difference method

FEM finite element method

FIML field inversion and machine learning

FOM full order model

FVM finite volume method

GCN graph convolutional network

GNN graph neural network

IC initial condition

LCO limit cycle oscillation

LES large eddy simulation

MGN MeshGraphNet

ML machine learning

MLP multi-layer perceptron

MPE missing point estimation

MPNN message passing neural network

MSE mean squared error

ODE ordinary differential equation

PDE partial differential equation

PINN physics-informed neural network

POD proper orthogonal decomposition

RANS Reynolds-averaged Navier-Stokes

RBE reduced basis element

RBF radial basis function

RCM reduced-complexity model(ing)

RD-MD reduced-domain training and multi-domain prediction

RK Runge-Kutta

RMSE root mean square error

RNN recurrent neural network

ROM reduced order model(ing)

SDF signed distance function

SISS sub-iteration serial staggered

SR super-resolution

SVD singular value decomposition

TCAE temporal convolutional autoencoder

TCN temporal convolutional network

UQ uncertainty quantification

WM-LES wall-modeled large eddy simulation

ABSTRACT

Many-query scientific and industrial applications, such as design, demand affordable yet accurate computational models. Data-driven Reduced-Complexity Models (RCMs), which are the focus of this thesis, typically require a significant amount of training data. Despite efficiency gains in the prediction stage, the large expense in the generation of the training data contradicts the low-cost pursuit of a RCM, and can make the *creation* of the RCM unaffordable. With this motivation, this thesis proposes a number of techniques towards the end of addressing the development of RCMs in what we refer to as the *low data regime*, which can manifest itself in three ways in practical problems: 1) Spatially, high-fidelity simulations can only be conducted on a truncated sub-domain, or for an individual component of a larger system; 2) Temporally, only a short period of dynamics can be simulated, spanned by a limited number of snapshots; and 3) Parameterically, only a limited set of operating conditions can be covered in training.

As a motivating example, this thesis considers a rocket engine combustor, for which a direct full-system simulation is not affordable. In concert with the underlying physics and design requirements, the computational domain is decomposed into 1) individual injectors and their neighboring downstream regions, which share a few types of identical geometries, yet experience computationally expensive reacting flow dynamics; and 2) the further downstream chamber and nozzle, which experiences relatively simple acoustics-dominated dynamics, yet can be connected with different numbers and arrangements of injectors. The responses to different upstream config-

urations, and possible consequential changes in the computational grid and even the geometry cannot be addressed by current data-driven RCMs. Our approach uses a RCM for the former, and an inexpensive PDE solver for the latter, such that the efficiency of a RCM and the flexibility of a PDE solver are utilized at the same time. Ideally, the training of the RCM only depends on independently-generated solutions for a single injector region, and no full-system full-order simulation is required. The realization of the strategy depends on three key aspects: 1) reduced-domain training – training data generation in an isolated reduced-domain, 2) model design – design of efficient and predictive RCMs, and 3) model-solver coupling – effective interfacing methods to couple a data-driven RCM and a high-fidelity solver.

In the reduced-domain training aspect, the truncation interface of an isolated reduced-domain is treated as a characteristic boundary, on which perturbation signals are imposed to represent the possible external responses from a coupled domain. In the model-design aspect, a series of techniques and integrated models are proposed in response to common limitations of existing methods. The Convolutional Autoencoder (CAE) is introduced as a nonlinear and more effective dimensionality reduction method in contrast to Proper Orthogonal Decomposition (POD) in Reduced Order Models (ROMs). Autoregressive deep-learning models are introduced to replace the expensive intrusive computation of nonlinear residuals, and form a family of non-intrusive ROMs in combination with either a POD or a CAE. In pursuit of even higher efficiency, a model based on multi-level autoencoder networks is proposed, which predicts the entire spatio-temporal field at unseen parameters in a "one-shot" manner. To avoid the geometry dependency generated by global dimensionality reduction, local surrogate models such as Convolutional Neural Networks (CNNs) and Graph Neural Networks (GNNs) are studied. For the latter in particular, mesh-to-graph mapping procedures, including boundary treatments, are introduced for different types of spatial discretization methods, which enables a model to pro-

cess irregular data structures widely used in scientific computing. To alleviate the brute-force fitting of high-order terms, and naive concatenation of hierarchical, heterogeneous features, the idea of Conditional Parameterization (CP) is utilized. CP is generalized to transform network weights into learnable functions of parameters, which is shown to result in significant drop-in improvements in a variety of tasks. In combination with a GNN, physical quantities, as well as graph edge features, are used to parameterize the weights of node features, and the resulting Conditionally Parameterized Graph Neural Network (CP-GNet) model is shown to be predictive on complex reacting flows and unseen irregular meshes.

Two major challenges in coupling RCMs with PDE solvers are: 1) incorporating physical interface conditions into non-intrusive ROMs that process non-physical reduced order variables, and 2) coupling a RCM with a high-fidelity PDE solver that uses a finer time-step and multiple sub-iterations. To address the former, a number of interface coupling strategies are realized. For the latter, the existing Conventional Serial Staggered (CSS) procedure is discussed and specifically improved for the RCM-PDE solver coupling context, leading to the Sub-Iteration Serial Staggered (SISS) procedure. The aforementioned techniques in the three key aspects are validated in both a stand-alone and integrated fashion on a hierarchy of test problems, beginning with the Burgers equation. In the final test case of a single-injector rocket engine chamber, a CP-GNet is independently trained for the upstream physics-intensive sub-domain with limited data, and coupled with a solver for the downstream variable chamber using SISS. The realization of a single-interface framework is shown to be predictive for unseen testing geometries.

The data-driven RCMs proposed in this work are not restricted to specific applications, except for the need to adapt the inputs and outputs. For predictions on a fixed geometry, non-intrusive ROMs provide remarkable efficiency improvements, especially when the time period of interest is also fixed across different parameters such

that the multi-level autoencoder network is viable. Local surrogate models eliminate the dependence on the global spatio-temporal domain, and generalize to more applications. With the help of interface models, the model-solver coupling approaches are also shown to generalize to different types and schemes of models and solvers. The resulting multi-domain framework endows significantly more flexibility to the applications of ROMs, yet still requires a fixed geometry in the ROM sub-domain. For more generalizable models such as the CP-GNet, the framework also reduces the training effort and improves the prediction performance. Despite the encouraging results, error estimation and stability properties require formal quantification, and remains an open topic for further research.

CHAPTER I

Introduction

1.1 Numerical Simulations and the Need for Reduced-Complexity Modeling

Numerical experiments are critical in many scientific and industrial problems that are risky, hard to measure, or expensive, such as nuclear reactions (*Khan et al.*, 2017; *Becker and Laurien*, 2003), nano-scale processes (*Sheikholeslami*, 2017; *Kundrapu and Keidar*, 2012), hypersonic flows (*Fu et al.*, 2018; *Zhong and Wang*, 2012), and rocket engines (*Harvazinski et al.*, 2015; *Urbano et al.*, 2017). The core processes of such experiments are the numerical simulations that enable the virtual representation and evolution of physical systems. They are also used as an indispensable tool to verify theories and mathematical models through digital reproductions of real-life observations, such as social interactions (*Vallacher et al.*, 2017), gravity waves (*Remmler et al.*, 2013), and biochemistry processes (*Lane et al.*, 2013). Accurate and predictive simulations benefit the public and the planet every day in the forecasting for weather (*Schalkwijk et al.*, 2015), climate (*Fuhrer et al.*, 2018), and pollution propagation (*Prants et al.*, 2011).

Although the quantities of interest involved in a numerical simulation, e.g., pressure, velocity, temperature, time, etc., are continuous in nature, their solutions have

to be completed in a discrete manner, in order to represent infinite Degrees of Freedom (DoFs) using finite ones. The fidelity of simulations increases with the resolution of the discretization, together with the cost. Taking the methods in the Computational Fluid Dynamics (CFD) field for example, the highest-fidelity method, the Direct Numerical Simulation (DNS) (*Moin and Mahesh*, 1998; *Lee and Moser*, 2015; *Domingo et al.*, 2005) solves the complete Navier-Stokes equations without approximations, and requires a resolution finer than the smallest length scale for the eddies in turbulent flow, the Kolmogorov scale. Denoting the streamwise characteristic Reynolds number by Re_L, the required number of discrete grid points by DNS is estimated to be $Re_L^{37/14}$ (*Choi and Moin*, 2012), which translates to more than 10^{15} points for an average airplane. The Large Eddy Simulation (LES) (*Smagorinsky*, 1963; *Lilly*, 1966; *Lee and Cant*, 2017) relaxes the requirement on the grid resolution by using a sub-scale model for the dynamics smaller than the grid size. A resolution that can resolve 80% to 90% of the turbulence kinetic energy is suggested in practice (*Pope*, 2004; *Matheou and Chung*, 2014), which effectively reduces the required number of grid points to $Re_L^{13/7}$ (*Choi and Moin*, 2012). However, this is still prohibitively expensive for industrial applications in the decades to come (*Jameson*, 1999; *Slotnick et al.*, 2014). Based on the fact that the small eddies consuming the most grid points are prevalent in the near-wall regions, models of even lower fidelity, such as the Reynolds-Averaged Navier–Stokes (RANS) models (*Baldwin and Barth*, 1991; *Spalart and Allmaras*, 1992; *Wilcox et al.*, 1998; *Launder and Spalding*, 1983) are used to replace the LES in these regions, and form a hybrid model called Wall-Modeled LES (WMLES) (*Piomelli and Balaras*, 2002; *Choi and Moin*, 2012; *Bose and Park*, 2018), which is the current state-of-the-art approach for full vehicle simulations (*Slotnick et al.*, 2014; *Goc et al.*, 2020). RANS models are also often used independently due to their affordable cost. Similar low-fidelity approximations are common in other fields, such as the reduced chemical kinetic mechanisms for combustion (*Ra and Reitz*, 2008; *Gou et al.*, 2010).

In many-query scenarios such as design, computations are cost-sensitive both resource- and time-wise, and therefore, trade-offs between the fidelity and the efficiency become inevitable. Classic low-fidelity options such as RANS, however, risk losing a considerable amount of detail, and are not reliable in problems where it deviates from canonical configurations (*Tinoco et al.*, 2005; *Slotnick et al.*, 2014). As a consequence, Reduced-Complexity Modeling (RCM), including Reduced Order Modeling (ROM) and other data-driven surrogate modeling methods, have emerged as a promising future direction for research. In these approaches, an offline stage aims to extract problem-specific information (such as a projection basis or a latent space) such that inexpensive computations can be performed during the online predictive stage. Among existing approaches, non-intrusive methods, including the recently emerging deep learning models (neural networks), have demonstrated impressive advantages in computational cost reduction and robustness. However, such methods discard the governing equations of the original dynamical system, thus face challenges dealing with different types of boundary conditions, which are commonly formulated as constraints to these equations. The generation of the prerequisite training data itself can also be expensive, which places a low data limit on the training process. Moreover, most neural networks are originally designed for tasks such as computer vision and natural language processing, and are migrated to scientific computation fields with minor modifications. The designs do not take physical relationships, or the often irregular and non-uniform spatial discretization into consideration, which imposes intrinsic limitations on the efficiency and flexibility of their applications to numerical simulations. This work is conducted with the hope that it can be of value to these challenges.

1.2 State-of-the-Art Reduced-Complexity Models

RCMs and other data-driven models, even restricted to the CFD field, have countless flavors and applications. Broad applications include mesh optimization, parameter planning, model augmentation, surrogate modeling, flow control, uncertainty quantification, and flow mechanism interpretation, etc. In this section, we provide an overview of the current state-of-the-art RCMs directly used in the solution-computation process. Based on the role of the model in the process, the approaches are classified into four categories, namely:

1. Model augmentation and inference, where the RCM output is an improved set of parameters for an existing low-fidelity model, or a direct replacement for the latter.

2. Coarse-graining solver acceleration, where the RCM learns an efficient high-order numerical scheme, or performs coarse-graining as an appended step to the existing solver.

3. Reduced order modeling, where a low-cost approximated solution is performed on a low-dimensional subspace.

4. Surrogate modeling, where the RCM replaces the most expensive component(s) of a solver, or the entire solver, and directly outputs quantities of interest.

A shared goal of the models is to reduce the total computational complexity in achieving high-fidelity solutions. It should be pointed out that although arranged in a hierarchical order, there is no definite relationship between the complexities of the approaches.

1.2.1 Model Augmentation and Inference

Even before the recent rise of data science, traditional turbulence modeling approaches had already been relying heavily on data for the manual validation, correction, and calibration of empirical formulations and their parameters. With the complexity limited by the processing power of any human researcher, a traditional turbulence model either is a global one for all flow conditions and regions, or has a limited number of parameter sets, corresponding to a few simple representative types of flow, such as a channel or pipe flow. In practice, the choice of model has to be made based on a review of different models on a comparable problem (*Yusuf et al.*, 2020).

In the past decade, with the help of Machine Learning (ML) techniques, the portability and accuracy of turbulence models have been largely improved. In pioneering work (*Oliver and Moser*, 2011; *Dow and Wang*, 2011), Bayesian Uncertainty Quantification (UQ) approaches are used to describe the discrepancy in model-predicted terms as Gaussian random fields instead of scalars. *Tracey et al.* (2013) used kernel regression to train a local error model that describes the discrepancy as a probabilistic function of local features from a low-fidelity solution. Similar UQ approaches were validated on richer feature sets and more complex ML models (*Xiao et al.*, 2016b; *Vollant et al.*, 2014; *Wang et al.*, 2017). For predictive tasks, the Field Inversion and Machine Learning (FIML) family of approaches (*Duraisamy et al.*, 2015; *Tracey et al.*, 2015; *Singh et al.*, 2017c; *Parish and Duraisamy*, 2016; *Srivastava and Duraisamy*, 2022) were developed, which enforces consistency between the target output data and the optimal prediction from the baseline low-fidelity model, and effectively improved the model robustness in a predictive setting. FIML models have been successfully applied to a wide range of flow conditions and baseline models (*Singh et al.*, 2017a,b; *Holland et al.*, 2019; *Sirignano et al.*, 2020).

In addition to augmenting existing low-fidelity models, ML is also applied to infer

a full closure model. *Schmelzer et al.* (2020) performs a symbolic regression to learn the optimal model-form from a library of candidates. A more popular approach is to train an equation-free model in a "purely" data-driven manner. Point-wise closures are learned using simple fully-connected neural networks in *Maulik et al.* (2019); *Subel et al.* (2021); *Xie et al.* (2020). Quantities within a local neighborhood of the flow field are included in a richer set of input features using Convolutional Neural Networks (CNNs) in *Yuan et al.* (2020) to bring potentially higher accuracy and robustness. *Um et al.* (2020) introduced an interactive training method, where the low-fidelity solver is integrated into the training loop for its correction term, and demonstrated significant performance improvement over traditional offline training methods.

1.2.2 Coarse-Graining Solver Acceleration

While being more accurate than baseline turbulence models, the augmented or inferred models do not bring improvements to the computational efficiency. In pursuit of the latter, ML-enabled coarse-graining is being widely used. Different from a closure/turbulence model in the previous level that aims to minimize the gap between a low-fidelity model and the projected solution from a high-fidelity model on the same coarse grid, a coarse-graining model provides a high-order accuracy solution (spatially, such as applying a non-zero gradient within a discretized volume) on a coarse grid to match the low-order accuracy solution on a fine grid. It is hard to determine when was the first exploration on using ML in this direction, but it is clear that a few early inspirations were drawn from the image Super-Resolution (SR) models such as the SRCNN (*Dong et al.*, 2015) and SRGAN (*Ledig et al.*, 2017) in the computer vision field. *Fukami et al.* (2019a) examined the possibility of applying variants of the SRCNN for flow-field SR and demonstrated promising results on both laminar and turbulent flows. More tailored SR approaches towards CFD applications have also been proposed, such as separating different velocity components (*Guo et al.*, 2020), or

predicting high-order Discontinuous Galerkin (DG) coefficients (*Pradhan et al.*, 2020; *Pradhan and Duraisamy*, 2021).

For predictive tasks, *Obiols-Sales et al.* (2020) used a CNN to bypass a large number of converging iterations in the solution of steady problems. *Bar-Sinai et al.* (2019) used neural networks to learn the interpolation coefficients for discretization schemes, and achieved comparable performance to existing mathematically derived high-order schemes on simple 1D problems. The approach was further validated on 2D turbulent flows in *Zhuang et al.* (2020). *Kochkov et al.* (2021) applied the data-driven discretization together with a data-driven closure term, and outperformed traditional baselines in both accuracy and efficiency. *Pathak et al.* (2020) appended an upscaling-processing-downscaling CNN to an existing solver as an explicit coarse-graining step. A similar approach is taken in *Belbute-Peres et al.* (2020), with the CNN replaced by a more flexible Graph Neural Network (GNN).

1.2.3 Reduced Order Modeling

1.2.3.1 Projection-Based Dimensionality Reduction

ROMs, based on projection-based dimensionality reduction methods, are among the most popular applications of machine learning in computational physics. Such methods construct a low-dimensional subspace, on which the original high dimensional system is approximated by a set of reduced order variables. ROM computations are then performed on the reduced order variables instead of the full system, which leads to a significant decrease in the DoFs to be solved. Projection-based ROMs commonly use a linear subspace constructed using the Proper Orthogonal Decomposition (POD) (*Berkooz et al.*, 1993; *Rowley et al.*, 2004). Other linear basis construction methods include balanced truncation (*Moore*, 1981; *Safonov and Chiang*, 1989), reduced basis methods (*Peterson*, 1989; *Prud'Homme et al.*, 2001; *Rozza et al.*, 2007), rational interpolation (*Baur et al.*, 2011), and proper generalized decom-

position (*de Almeida*, 2013; *Berger et al.*, 2016). Linear basis ROMs have achieved considerable success in complex problems such as turbulent flows (*Rowley*, 2005; *Carlberg et al.*, 2011; *Hijazi et al.*, 2020) and combustion instabilities (*Huang et al.*, 2018; *Xu et al.*, 2019). Recently, a special type of neural network, the autoencoder (*DeMers and Cottrell*, 1993), has been used to replace the POD in ROMs. An autoencoder performs data compression and decompression through stacked nonlinear operations, which can be regarded as a projection process onto nonlinear manifolds. With autoencoders, significant improvements on the compression rate and prediction accuracy over POD-based models have been demonstrated on multiple problems (*Carlberg et al.*, 2019; *Lee and Carlberg*, 2020; *Murata et al.*, 2020; *Xu and Duraisamy*, 2020), in the cost of a higher model complexity, and sometimes additional requirements on the input-data structure, e.g. Euclidean data for convolutional autoencoders.

1.2.3.2 Intrusive ROMs

Independent of the type of basis that is employed, ROM approaches can be broadly categorized into *intrusive* and *non-intrusive* methods. Intrusive ROMs perform a formal projection of the original governing equations onto the reduced dimensional manifold using Galerkin (*Rowley et al.*, 2004; *Couplet et al.*, 2005) or Petrov Galerkin (*Carlberg et al.*, 2011; *Parish et al.*, 2020) formulations. In nonlinear systems, however, intrusive ROMs require additional constructs for efficiency. Although it has been shown that acceleration can be achieved courtesy of the decreased stiffness of the ROM compared to the FOM (*Marley et al.*, 2015; *Amsallem and Farhat*, 2011), no reduction of the computing cost for the nonlinear terms is realized within each time-step because the full residual will still have to be computed before projection to the reduced space. Sparse acceleration methods such as the Missing Point Estimation (MPE) (*Astrid*, 2004; *Astrid et al.*, 2008) and the Discrete Empirical Interpolation Method (DEIM) (*Chaturantabut and Sorensen*, 2009; *Drmac and Gugercin*, 2016)

have been developed to reduce the cost by restricting the computation of nonlinear terms to a subset of the state variables in reduced order systems. However, these methods require further development and are reported to suffer from accuracy and numerical instability problems, especially when applied in a predictive stage (*Huang et al.*, 2018) or when noise is present (*Argaud et al.*, 2017). Recently, adaptive (*Peherstorfer and Willcox*, 2015; *Zimmermann et al.*, 2018) and oversampling methods (*Peherstorfer et al.*, 2020) have been introduced to stabilize DEIM methods and to provide improved predictive capabilities. It has to be mentioned that effective sampling approaches have not been developed for nonlinear manifolds.

1.2.3.3 Non-Intrusive ROMs

Non-intrusive methods bypass the governing equations, and compute the reduced order variables using a second level of RCMs (the POD/autoencoder being the first level). Nearly 20 years ago, *Milano and Koumoutsakos* (2002) used a fully-connected neural network to predict the POD coefficients for near-wall turbulent flows. For transient problems, the most popular choice is the autoregressive DL models, such as the Recurrent Neural Networks (RNNs) (*Rumelhart et al.*, 1988; *Gonzalez and Balajewicz*, 2018; *Maulik et al.*, 2021), Long Short-Term Memory networks (LSTMs) (*Hochreiter and Schmidhuber*, 1997; *Mohan et al.*, 2019; *Maulik et al.*, 2020), Temporal Convolutional Networks (TCNs) (*Oord et al.*, 2016; *Xu and Duraisamy*, 2020). These methods are discussed in more detail in Chapter III.

1.2.4 Surrogate Modeling

The aforementioned ROMs and other solver acceleration methods require the assistance of a companion computation, either in the form of a projection or conducted with the existing solver. Seeking for an even lower computation cost, an end-to-end surrogate model is a single model that directly outputs the quantities of interest over

the computation domain.

1.2.4.1 Convolutional Neural Networks (CNNs)

As the name suggests, a CNN refers to a network that involves convolution computations. The most basic element of a CNN is a convolution kernel, which is a set of trainable parameters arranged in a fixed small spatial frame, such as a 3×3 square, that is used to swipe through the discrete spatial points. For each centering point during the swiping process, the output is computed by a local convolution between the kernel parameters and the features on the points covered inside the kernel frame. Multiple such kernels are used to swipe through the discrete spatial points. Multiple parallel and serial stacks of such kernels can be used in a single CNN. Due to the fact that the number of parameters in the convolution kernels is independent of the domain size, CNNs have long been the most popular choice for processing large-scale spatially distributed data. In early applications of CNNs for full-field predictions, computations are performed in a one-shot manner for either derived statistic quantities such as the pressure coefficients (*Yilmaz and German*, 2017) or steady flows (*Guo et al.*, 2016; *Sekar et al.*, 2019), which usually requires a lower model complexity and training effort than a full spatio-temporal system. For applications not sensitive to the point-wise accuracy, CNNs are also used to generate qualitatively realistic unsteady flows (*Kim et al.*, 2019; *Fukami et al.*, 2019b).

Tompson et al. (2017) took a pioneering step towards an accurate transient full-field surrogate model by using a CNN to replace the expensive pressure computation for incompressible flows, with the velocity computation retained in a traditional solver. More recently, end-to-end transient surrogate models for a complete set of variables have been developed successfully (*Kim et al.*, 2019). *Qin* (2020) used a flattened concatenation of the variables within a local neighborhood as the input to a fully connected network. The local selection and the concatenation of inputs reproduces

the essence and benefits of a CNN on the simpler network architecture. There are also convolutional models that are able to process spatial and temporal dimensions simultaneously, such as the ConvLSTM (*Xingjian et al.*, 2015) and the spatio-temporal convolution (*Yu et al.*, 2017), which have been successfully applied to popular deep learning tasks such as video generation. However, complex spatio-temporal systems in scientific computing often exhibit a much larger data size per unit time, therefore such models cannot be easily applied. *Xu and Duraisamy* (2020) introduced a multi-level convolutional approach that efficiently encodes the full spatio-temporal domain, and realized parametric prediction for unsteady flows on different Reynolds numbers and inflow angles.

1.2.4.2 Graph Neural Networks (GNNs)

Despite promising results on canonical problems, an outstanding limitation of CNNs is that they can only work with problems discretized on a Euclidean space, represented by a uniform linear/rectangular/cuboid mesh, which largely restricts its application on complex geometries and multi-scale physics. Additional cost and error are introduced when existing data is interpolated from a non-uniform grid onto a uniform one and back. Encoded geometric labels e.g. Signed Distance Function (SDF) (*Guo et al.*, 2016; *Bhatnagar et al.*, 2019), and more mathematically formal treatments e.g. elliptic coordinate transformation (*Gao et al.*, 2021) have also been used to alleviate this restriction. However, they are still bound to meshes with quadrilateral cells and require additional pre-processing of the geometric information, which is often difficult to be automated.

The Graph Convolutional Network (GCN) (*Kipf and Welling*, 2016) and more generally, the GNNs, work directly on graph representations of non-uniform grids, and thus become a natural alternative to CNN for CFD applications. Multiple architectures in the family of GNNs have shown successes in processing irregular,

non-Euclidean features. Applications include cloud classification (*Landrieu and Simonovsky*, 2018; *Wang et al.*, 2019b), action recognition (*Yan et al.*, 2018) and control (*Sanchez-Gonzalez et al.*, 2018), traffic forecasting (*Yu et al.*, 2017; *Zhang et al.*, 2018), quantum chemistry (*Gilmer et al.*, 2017). Most GNNs used as end-to-end surrogate models fall into a more specific sub-category, the Message Passing Neural Network (MPNN) (*Gilmer et al.*, 2017), which treats graph convolutions as messages passed between nodes through edges. Initial attempts were made on (*Li et al.*, 2018; *Sanchez-Gonzalez et al.*, 2020). *Li et al.* (2018) made an initial attempt on particle-based systems. The Graph Network-based Simulators (GNS) (*Sanchez-Gonzalez et al.*, 2020) introduced a tailored encoder-processor-decoder structure for simulations. MeshGraphNets (*Pfaff et al.*, 2020) extended the structure to mesh-based simulations, and demonstrated impressive performance on a wide range of physical systems.

1.2.4.3 Physics-Informed Neural Networks

A substantial difference between tasks in the computational physics field and other popular machine learning tasks is the availability of governing equations, usually in the form of PDEs. In a Physics-Informed Neural Network (PINN) (*Raissi et al.*, 2019), the violation of physics, quantified by the residual of the governing PDEs, is included as a regularization term in the loss function in the training process. PINNs have been shown to sufficiently reduce the required amount of labeled training data (*Raissi et al.*, 2019). With proper treatment of the boundary and initial conditions, even data-free training can be realized (*Sun et al.*, 2020). It should be noted that PINNs are characterized by the training method instead of a specific network architecture. The idea has been successfully incorporated into fully-connected networks (*Raissi et al.*, 2019; *Yang et al.*, 2019b), as well as CNNs (*Geneva and Zabaras*, 2020; *Gao et al.*, 2021).

1.2.4.4 Data-Driven System Identification

Compared with the aforementioned direct data-driven predictions, a more interpretable approach towards surrogate modeling is to exploit the latent data structures to infer ROM equations. Data-driven identification and solution of dynamical systems have a long history. As early as more than 40 years ago, neural networks had been used for autoregressive approximations for nonlinear responses (*Kumpati et al.*, 1990; *Narendra and Parthasarathy*, 1992). Close connections with PDE models were quickly made (*Dissanayake and Phan-Thien*, 1994; *Lagaris et al.*, 1998). *Brunton et al.* (2016) performed sparse system identification based on a pre-collected dictionary of nonlinear equations. Accurate and reliable surrogate solutions have already been realized for many paradigmatic dynamical systems, such as the FitzHugh–Nagumo model (*Kevrekidis et al.*, 2003), the Lorenz attractor (*Brunton et al.*, 2016), and the Kepler problem (*Qin*, 2020). *Gu* (2011) introduced the idea of reducing arbitrary nonlinearity to quadratic-bilinearity via lifting, which have been successfully applied to complex reacting flows (*Kramer and Willcox*, 2019b,a; *McQuarrie et al.*, 2021).

1.3 Overview of Model Development and Inference

1.3.1 Full Order Model

We consider time-evolving dynamic systems that can be represented by a PDE of the following form:

$$\begin{aligned} \frac{\partial \mathbf{q}(\mathbf{x},t,\mu)}{\partial t} + \mathbf{f}(\mathbf{q}(\mathbf{x},t,\mu),\mathbf{x}) &= 0, \\ \mathbf{x} \in \Omega, t \in [0, t_{\text{end}}], \\ \mathbf{q}(\mathbf{x},0,\mu) &= \mathbf{q}_0(\mathbf{x},\mu), \\ \mathbf{a} \odot \mathbf{q}(\partial\Omega,t,\mu) + \mathbf{b} \odot (\nabla \mathbf{q}(\partial\Omega,t,\mu) \cdot \mathbf{n}_{\partial\Omega}) &= \mathbf{f}_{\partial\Omega}, \end{aligned} \tag{1.1}$$

where $\mathbf{q} \in \mathbb{R}^{n_v}$ denotes the vector for the unknown state variables, t denotes the time, $\mathbf{x}$ denotes the spatial coordinates in domain Ω, μ is a parameter vector describing the operating condition of the system, $\mathbf{f}$ is a nonlinear operator that includes spatial derivatives and source terms, and the symbol $\odot$ denotes an element-wise vector multiplication. The pre-defined spatial field $\mathbf{q}_0(\mathbf{x})$ at $t = 0$ is called the Initial Condition (IC). The function $\mathbf{f}_{\partial\Omega}$ represents a set of constraints on the domain boundary $\partial\Omega$, which is called the Boundary Condition (BC). A BC is commonly defined on a linear combination of the boundary value $\mathbf{q}(\partial\Omega, t)$ and the boundary-normal gradients $\nabla\mathbf{q}(\partial\Omega, t) \cdot \mathbf{n}_{\partial\Omega}$, with $\mathbf{n}_{\partial\Omega}$ being the boundary-normal vector. A BC is categorized as Dirichlet when $\mathbf{a} = \mathbf{1}, \mathbf{b} = \mathbf{0}$, Neumann when $\mathbf{a} = \mathbf{0}, \mathbf{b} = \mathbf{1}$, and Robin for other combinations. In practice, different BCs can be applied to different parts of the domain boundary simultaneously.

In general, the numerical solution of Eq. (1.1) is performed in a discrete manner, where Ω is spatially discretized into n_x grid nodes or cells. The solution is represented by the corresponding nodal or cell-centered values at discrete time-steps. In a discrete setting, $\mathbf{q}$ denotes the set of unknowns at all discrete locations. We take the space index i as a subscript and the time index k as the superscript, such that $\mathbf{q}_i^k$ denotes the values for the i-th node/cell located at $\mathbf{x}_i$, at the k-th time-step.

The solution requires a spatial discretization of the operator $\mathbf{f}$, common approaches including the Finite Difference Method (FDM), the Finite Volume Method (FVM) and the Finite Element Method (FEM). After the spatial discretization, an ODE is obtained:

$$\begin{aligned} \frac{d\mathbf{q}}{dt} + \mathbf{f}(\mathbf{q}, t) = 0, \\ \mathbf{q}(0) = \mathbf{q}_0, \end{aligned} \tag{1.2}$$

which is called the *semi-discrete* form for Eq. (1.1). The IC for Eq. (1.2) can be directly inherited from that for Eq. (1.1) in a discrete-sampling manner. The imple-

mentation of different BCs, however, is less straightforward and depends closely on the specific type of spatial discretization applied. For example, a Dirichlet BC for a finite-volume cell immediately inside the domain boundary can only be imposed weakly by specifying the values for an adjacent "ghost cell" across the boundary.

Schemes for time-integrating Eq. (1.2) include methods such as the Runge-Kutta (RK) schemes and the Backward Difference Formula (BDF), etc. In this work, we focus on the development of RCMs. When no dimensionality reduction or fidelity reduction is performed, any combination of the spatial discretization and time-marching schemes for Eq. (1.2) will be regarded as a Full Order Model (FOM), and their solution is regarded as the ground truth for the data-driven RCMs. A RCM is called *intrusive* when the FOM scheme is accessed in the computation, and otherwise *non-intrusive*.

1.3.2 POD-Galerkin ROM

The POD-Galerkin ROM is an intrusive model achieved by performing the Galerkin projection of the FOM Eq. (1.2) using POD bases. It is probably the most basic, and also the most widely used type of RCM in numerical simulations. The procedures for developing a standard POD-Galerkin ROM are introduced below, which can be contrasted with the more complex model developments in the rest of the thesis.

Assume that the possible discrete FOM solutions span a high-order space $\mathcal{V}_q = \mathbb{R}^{n_q}$, where $n_q = n_x n_v$ is the total DoFs per time-step. Projection-based dimensionality reduction is achieved through a mapping from $\mathcal{V}_q$ to a low order subspace $\mathcal{V}_r = \mathbb{R}^{n_r}$. Then a ROM can be obtained by solving for a vector $\mathbf{r} \in \mathcal{V}_r$, as an approximation for $\mathbf{q}$, such that the total DoFs are reduced from n_q to n_r. The vector $\mathbf{r}$ is called a *reduced order variable*.

POD seeks a L2-optimal low rank representation of a data matrix using the Singular Value Decomposition (SVD) technique. To perform SVD, a training set of FOM data with n_t steps is collected as a *snapshot* matrix $\mathbf{Q} \in \mathbb{R}^{n_q \times n_t}$, for which each

column is the solution for one time-step:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{q}_1^1 & \cdots & \mathbf{q}_1^{n_t} \\ \vdots & \ddots & \vdots \\ \mathbf{q}_{n_x}^1 & \cdots & \mathbf{q}_{n_x}^{n_t} \end{bmatrix} \tag{1.3}$$

Considering real solutions only, the SVD of $\mathbf{Q}$ is:

$$\mathbf{Q} = \mathbf{U}\mathbf{\Sigma}\mathbf{Z}^T, \tag{1.4}$$

where the left singular matrix $\mathbf{U} \in \mathbb{R}^{n_q \times n_q}$ and the right singular matrix $\mathbf{Z} \in \mathbb{R}^{n_t \times n_t}$ are orthogonal matrices[1], and $\mathbf{\Sigma} \in \mathbb{R}^{n_q \times n_t}$ is a rectangular diagonal matrix, whose diagonal entries are the *singular values* of $\mathbf{Q}$. The columns of $\mathbf{U}$ can be regarded as linearly independent spatial modes for the FOM solution. By selecting the first n_r columns, a POD projection basis $\mathbf{V}$ is formed. When n_r is selected properly to be smaller than rank($\mathbf{Q}$), then $\mathbf{V}$ spans $\mathcal{V}_r$. The rest columns form a complementary basis $\mathbf{V}_\perp$ spanning $\mathcal{V}_\perp$, such that $\mathcal{V}_r \oplus \mathcal{V}_\perp = \mathbb{R}^{\text{rank}(\mathbf{Q})}$.

A L2-optimal approximation to $\mathbf{q}$ on the reduced space $\mathcal{V}_r$ is given by:

$$\mathbf{q} \approx \mathbf{V}\mathbf{r} = \sum_{l=1}^{n_r} \mathbf{r}_l \mathbf{V}_{:,l}, \tag{1.5}$$

where $\mathbf{r} = \mathbf{V}^T\mathbf{q}$.

Finally, Galerkin projection is conducted using the POD bases $\mathbf{V}$, and the resulting POD-Galerkin ROM equation is given by:

$$\frac{d\mathbf{r}}{dt} + \mathbf{V}^T\mathbf{f}(\mathbf{V}\mathbf{r}, t) = 0. \tag{1.6}$$

[1]The right singular matrix is more commonly denoted by $\mathbf{V}$, and $\mathbf{Z}$ is used here due to a conflict of notations.

1.3.3 Model Training

Model training refers to the process of fitting a model to a target input-output relationship. In data-driven approaches, the relationship is usually implied by an existing set of solutions, called training data. For linear models, the training is done with explicit computations, such as performing the SVD for the POD, or solving the linear regression equation. For nonlinear models, the training process consists of iterative updates for the model parameters to minimize a loss function that measures the gap between the target output and the model output.

1.3.3.1 The Minimization Problem

For regression problems, a common choice for the loss function is the Mean Squared Error (MSE). Between a target vector $\mathbf{y} \in \mathbb{R}^{n_y}$ and an approximation $\tilde{\mathbf{y}}$, the MSE is given by:

$$\text{MSE}(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{(\tilde{\mathbf{y}} - \mathbf{y})^T(\tilde{\mathbf{y}} - \mathbf{y})}{n_y}. \tag{1.7}$$

To alleviate *overfitting*, a penalty on the model parameters is added as a *regularization* term to the loss function. Common choices of regularization include the ℓ_1-norm and ℓ_2-norm. In this work, the latter is used. For a nonlinear model $\mathcal{F}(\mathbf{h}; \boldsymbol{\Theta})$ that takes an input vector $\mathbf{h}$ corresponding to the target output $\mathbf{y}$, with model parameters $\boldsymbol{\Theta}$, the final expression for the loss function is given by:

$$\mathcal{L}(\mathbf{h}, \mathbf{y}; \boldsymbol{\Theta}) = \text{MSE}(\mathbf{y}, \mathcal{F}(\mathbf{h}; \boldsymbol{\Theta})) + \lambda \left\| \boldsymbol{\Theta} \right\|_2, \tag{1.8}$$

where λ is a penalty coefficient. The optimal parameters $\boldsymbol{\Theta}^*$ can be expressed as:

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} \sum_{\{\mathbf{h},\mathbf{y}\} \in \{\mathbf{H},\mathbf{Y}\}} \mathcal{L}(\mathbf{h}, \mathbf{y}; \boldsymbol{\Theta}), \tag{1.9}$$

where $\mathbf{H}$ and $\mathbf{Y}$ are the training sets of input and output.

For neural networks, *Back propagation* is used to solve Eq. (1.9) based on the gradient $\nabla_{\boldsymbol{\Theta}}\mathcal{L}$. At an optimization step k, the update to the parameter can be expressed in a general form as:

$$\boldsymbol{\Theta}^{k+1} = \boldsymbol{\Theta}^{k} - \eta\mathcal{M}(\nabla_{\boldsymbol{\Theta}}\mathcal{L}^{k}), \tag{1.10}$$

where η is the learning rate, and $\mathcal{M}$ is a function that corrects the gradient with momentum terms. There are numerous types of such functions, and the Adam optimizer (*Kingma and Ba*, 2014) is used in this thesis.

1.3.3.2 Feature Scaling

Scaling is necessary for the development of data-driven models to avoid over-fitting to the features with larger magnitudes. For example, in common fluid problems, the measurement of pressure in Pascal is often several orders larger than that for temperature in Kelvin, and will dominate the parameter optimization process without proper scaling. In this thesis, each physical quantity or each digit in a reduced order variable vector is regarded as an individual feature, and is scaled independently. Unless otherwise specified, the standard deviation scaling is used by default. For a feature $\mathbf{h}$, the scaling is given by:

$$\mathbf{h}^{*} = \frac{\mathbf{h} - \mu(\mathbf{h})}{\sigma(\mathbf{h})}, \tag{1.11}$$

where $\mathbf{h}^{*}$ is scaled feature, and $\mu(\cdot)$ and $\sigma(\cdot)$ denotes the spatio-temporal mean and standard deviation, respectively.

1.3.4 Model Inference and Noise Injection

Model inference, or testing, refers to the process of computing quantities of interest based on unseen data (not included in the training process). For surrogate modeling of

a transient unsteady flow, a *rollout prediction* is required, which consists of recurrent model inference steps – the output from a leading step is used in the input for the following step. Inevitably, the inference error introduced will be carried out to the next, and consequentially all the future steps. This accumulation of error can quickly lead to unrealistic values and "blow-up" the prediction process. To alleviate the problem and enable robust long-term prediction, the training noise injection technique is commonly used. In this technique, an assumption is made that the inference error can be modeled by a random Gaussian noise field. In the training process, such a noise is added to the input in order to make the model robust against the accumulative error and compensate for it.

In this thesis, the surrogate model predicts the increment between two consecutive steps of the same variable(s). Without feature scaling and noise injection, the target model is given by:

$$\mathcal{F}\left(\mathbf{h}^k; \mathbf{\Theta}\right) = \Delta\mathbf{h}^k = \mathbf{h}^{k+1} - \mathbf{h}^k, \tag{1.12}$$

where the input $\mathbf{h}$ can be either a physical quantity or a reduced variable. In practice, the output and the input are independently scaled. After training noise injection and scaling using Eq. (1.11), the target model becomes:

$$\mathcal{F}(\frac{\mathbf{h}^k + \epsilon - \mu(\mathbf{h})}{\sigma(\mathbf{h})}; \mathbf{\Theta}) = \frac{\mathbf{h}^{k+1} - \mathbf{h}^k - \mu(\Delta\mathbf{h})}{\sigma(\Delta\mathbf{h})}, \tag{1.13}$$

where ϵ is a zero-mean Gaussian noise of the same shape as $\mathbf{h}$. The standard deviation of ϵ is set to be close to that for the single-step inference error in $\mathbf{h}$ through *a priori* tests. It should be made clear that ϵ is only manually injected in the training process, and is supposed to be implicitly represented by the model error in the rollout prediction. Although can be obtained through a simple transformation from

Eq. (1.13), the expression for a complete online update step is provided for clarity:

$$\mathbf{h}^{k+1} = \mathbf{h}^k + \sigma(\Delta\mathbf{h}) \left(\mathcal{F} \left(\frac{\mathbf{h}^k - \mu(\mathbf{h})}{\sigma(\mathbf{h})}; \mathbf{\Theta} \right) + \mu(\Delta\mathbf{h}) \right), \tag{1.14}$$

For simplicity, the un-scaled form Eq. (1.12) is used instead of the complete form Eq. (1.14) in the rest of the thesis. All models are supposed to be trained and inferred with proper noise injection and feature scaling.

1.4 Manifestations of the Low Data Limit and the Need for Domain Decomposition

Generally speaking, RCMs work inside the interpolation range of training data in a broadly defined "hyper-parameter" space. For numerical simulations, critical hyper-parameters include magnitudes, dynamic patterns, discretization characteristics, geometries, external forcing, etc. To create a training set that covers a large-enough envelope along all of the dimensions, the number of training points grows exponentially. It should be reminded that behind each training point is not a scalar, but instead, a complete spatio-temporal solution obtained from a full order simulation at that combination of hyper-parameters. Considering the example costs provided in Sec 1.1, it is important for the models to learn along every hyper-parameter dimension efficiently on a small number of training points. However, this is not the case for existing RCMs.

Among the models of our interest, ROMs are probably the most efficient in the inference stage, but they face challenges in processing certain types of hyper-parameters. A shared problem in POD, autoencoders, and the resulting ROMs based on them, is that they are typically formulated to generate a fixed mapping between the geometric coordinates and the compressed digits. Through its full lifespan, a basic projection-based model is restricted to a single discretized geometry or very limited variants of

it through transformations such as the elliptic coordinate transformation (*Gao et al.*, 2021). On the other hand, there are many cases in which the computation-intensive complex physics that requires a less expensive model concentrates in a small section of the domain, rather than spanning the full system. In these cases, an intuitive way to combine the efficiency of a RCM and the flexibility of a FOM is to allocate different parts of the domain to their most suitable models, and perform full-system predictions in a coupled manner. This model-solver coupling idea has been successfully verified with intrusive ROMs and solvers with simple time-integration schemes (*Baiges et al.*, 2013; *Huang et al.*, 2016; *Xu et al.*, 2019). However, as will be discussed in Sec. 1.4.2, certain challenges remain to be addressed in the development of a generic model-solver coupling strategy.

Even for a fixed geometry, using a single global set of bases for the whole domain also limits the training efficiency and predictive capability of the ROM w.r.t. different dynamic patterns. The limit is especially obvious in advection-dominated problems – a global projection basis is informative for the regions that are passed through by the local dynamic structures in the training, and will not be predictive when the structures travel outside the regions, no matter how locally simple they are. While the situation can be improved to a certain degree by using adaptive bases (*Peherstorfer and Willcox*, 2015; *Peherstorfer*, 2020; *Amsallem and Farhat*, 2011), some of the fundamental challenges persist. In this case, domain decomposition is also applied in a purely ROM setting, leading to the Reduced Basis Element (RBE) family of approaches (*Maday and Rønquist*, 2002; *Iapichino et al.*, 2016; *Hoang et al.*, 2021), in which the ROM bases are truncated into, or individually built for, the component sub-domains. With multiple sets of bases, the dynamics in different regions of the domain can be generated and learned individually, resulting in a higher training efficiency. However, training multiple parallel ROMs introduces additional workloads to the workflow. Challenges in synchronizing the different sub-domains are also raised.

The situation is improved with local surrogate models such as CNNs and GNNs. The models are regarded as "local" because the convolution and message-passing operations are defined as kernels on a center spatial point and a small neighborhood around it. The local kernels bring a *weight-sharing* property to the models – when a kernel slides through the input coordinates, it can be regarded as a single model called repeatedly with independent inputs. In the training, each point in a spatially discretized field can be regarded as an individual training sample, which significantly reduces the training cost. The kernels also efficiently capture local features, thus providing better predictive performance especially for propagating small-scale dynamics.

However, in scientific computing practice, the local kernels face new challenges. The spatial discretization of the computation domain is often irregular, exhibiting significant changes in local characteristics such as the number of neighboring points, edge lengths, and directions. The challenges are even worse on domain boundaries/interfaces, where the points within the range of a single kernel can be heterogeneous. Actually, this leads our view to a common problem with existing neural networks, that they often ignore the hierarchical relations between heterogeneous features, and concatenate them into a single input vector. In a GNN/MPNN, this is reflected by frequent concatenations of the graph edge and graph node features. The learning of high-order terms also remains mostly unguided – even simple quadratic terms are often fitted via a number of hidden units in a brute-force manner. Indeed, current successful surrogate models such as the MeshGraphNets (*Pfaff et al.*, 2020) require a large amount of training data (thousands of trajectories) to be portable between meshes and dynamic patterns. Such problems can be potentially addressed by conditional parameterization, i.e. to make the network parametric to certain features, such as connectivity patterns (*Stanley et al.*, 2009), layer embedding (*Ha et al.*, 2016), mean image features (*Yang et al.*, 2019a). More related to the scope of this thesis, the Edge Conditioned Convolution (*Simonovsky and Komodakis*, 2017) makes

the weights for graph node features dependent on edge features. The CP-GNet (*Xu et al.*, 2021) applied conditional parameterization to the encoder-processor-decoder structure with an extended choice of parameters including physical quantities and discretization characteristics, and successfully predicted a reacting flow in a complex geometry.

1.4.1 An Example Full-System Modeling Strategy.

In practice, the challenges imposed by different types of hyper-parameters can be intricately coupled, and the low data limit can be more severe. We take the study of a rocket engine combustor shown in Fig. 1.1 for example. As outlined in Fig. 1.1a, fuel and oxidizer from upstream manifolds are injected, mixed, and ignited through injector elements on an injector plate. Downstream to the plate is a combustion chamber, in which the gases further react and finally exit from a choked nozzle. In practice, a hurdle in the development of such engines is the combustion instability caused by the in-phase coupling between acoustic pressure oscillations and heat release, both of which can experience destructively high unsteady amplitudes, and lead to catastrophic engine failures. Within each injector element, the coupling is characterized by the disruption and resumption of fuel injection as pressure waves pass through, and consequential in-phase oscillations in the heat release. Extending to the entire multi-injector engine, the instability is further complicated by transverse modes, which create even greater amplitudes and require careful treatments, particularly an optimized design of the injector plate configuration (*Young*, 1995; *Harrje and Reardon*, 1972; *Harvazinski et al.*, 2019).

In the real world, the geometry requires hundreds of injectors. In an ideal and complete computer-aided design and engineering process, the exact number and arrangement of these injectors should be determined iteratively through multiple full-combustor simulations. However, even a coarsely-resolved simulation of the simplified

9-injector geometry in Fig. 1.1c took around 10 million CPU hours (*Harvazinski et al.*, 2019), and thus a completely simulation-based approach would be impossible. Unfortunately, traditional ROMs, and many other RCMs such as the PINN lack portability between different geometries. Even provided with such portability, a naive training process would require multiple full-system training runs to be predictive for new geometries, which would still be meaningless for a practical design purpose.

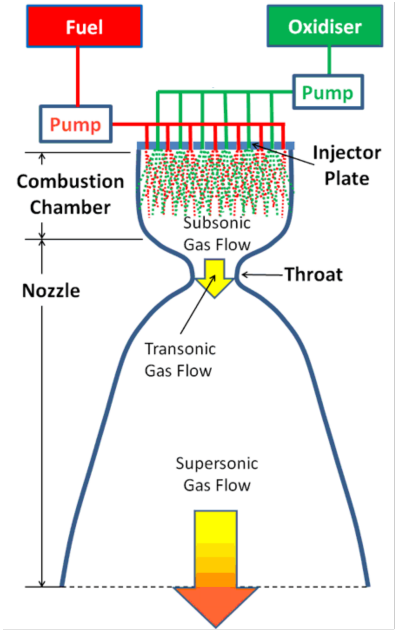
To improve the situation, a modeling strategy as outlined in Fig. 1.2 can be used. This strategy is based on two observations/assumptions:

1. The small-scale dynamics and detailed combustion processes that require the highest resolution and consequently the largest computational effort concentrate in the injectors and their neighboring downstream regions, as marked in Fig. 1.1c, whereas the large-scale dynamics in the further downstream chamber and nozzle can be simulated at an affordable cost.

2. The dynamics between these computation-intensive regions are statistically similar. This is because that they have identical geometries and inlet conditions, and interact with similar downstream responses that are dominated by the same group of pressure waves. As visualized in Fig. 1.1c, despite phase differences, the magnitude and frequency of the downstream responses only vary in a small range between the regions.

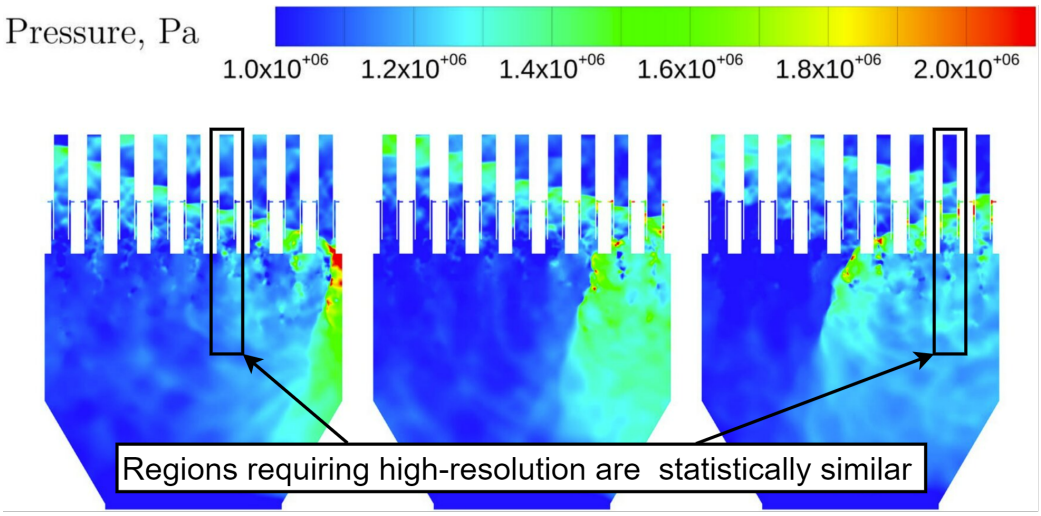
Based on the above, it is reasonable to expect that an efficient single-injector model can be trained, and mirrored for the required number of injectors in a full-system prediction, in combination with an inexpensive downstream solver that handles the variable geometry.

However, the mirrored models are not simple replications of the outputs from a single model – their downstream responses can be largely asynchronous due to span-wise dynamics in the chamber, which can be visualized by the phase differences



(a) Engine overview (*Lawson*, 2015).

(b) Real-world geometry: hundreds of injector posts in the injector plate of a Russian RD-170 engine (*Haeseler and Haidn*, 2017).



(c) Simulation geometry taking 10 million CPU hours: 9 injectors (*Harvazinski et al.*, 2019).

Figure 1.1: Example rocket engine combustor.

between different injectors in Fig. 1.1c. Therefore, the model is required to communicate effectively with the downstream domain retained in a CFD solver. It also has to be predictive under different downstream response patterns. The training of such a model is especially challenging as even a full-system FOM simulation is unaffordable, and a single run with a fixed number of injectors is not guaranteed to be representa-

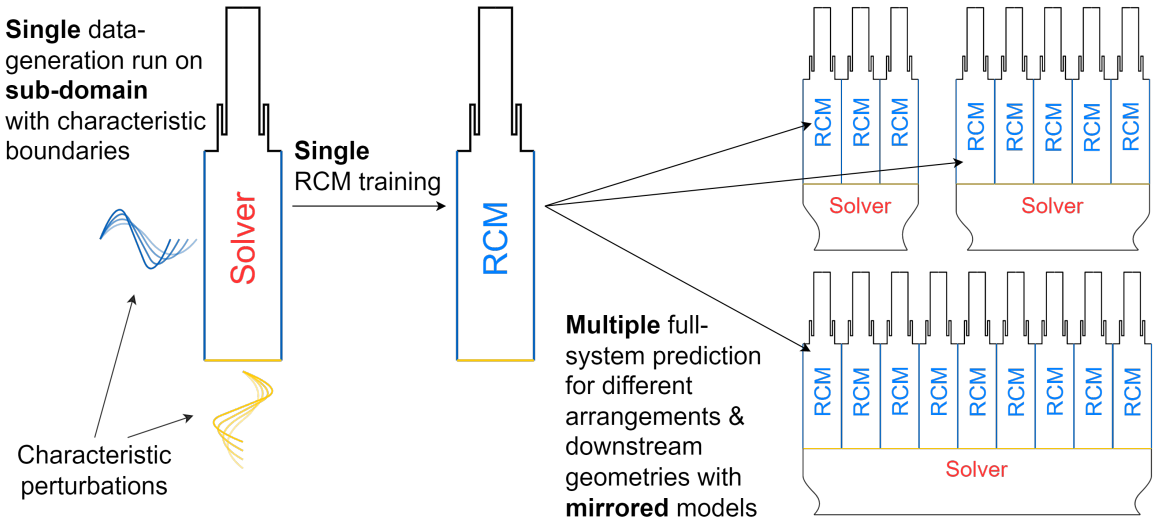


Figure 1.2: Example full-system modeling strategy.

tive for other configurations. In a series of papers (*Huang et al.*, 2016, 2017; *Xu et al.*, 2019; *Huang et al.*, 2019a), a multi-fidelity framework specialized for efficient ROMs for a decomposed domain has been proposed and studied. In this framework, the training data for the ROM sub-domain is generated without coupling with the rest of the system. Instead, characteristic perturbations are applied at the decomposition interface, which is designed to be representative of the possible responses from the pre-assumed FOM partner to be used in the coupled online prediction. This framework is shown to effectively reduce the number of training runs required for the prediction of a quasi-1D combustion system with a variable-length downstream chamber, but remains to be tested on more complex problems or non-intrusive models. *Huang* (2022) further validated the framework with intrusive ROMs for a 2D single-injector combustor, and the test case is also adopted in this thesis. The strategy in Fig. 1.2 extends this framework to multi-injector cases, with characteristic perturbations added to both the downstream boundary and the span-wise boundaries in the training run for the model sub-domains. This work contributes to the realization of this strategy.

1.4.2 Challenges in Model-Solver Coupling

It is not hard to find successful applications of domain-decomposition RCMs, in either a model-model coupling setting or a model-solver one. In the former, most works fall into the aforementioned RBE family of methods, and recent explorations focus on improving the compatibility between sub-domains projected to different sets of bases, such as the work *Iapichino et al.* (2016) and *Hoang et al.* (2021). In the model-solver coupling setting, intrusive ROMs have been used almost exclusively. For the full-system modeling strategy of our interest, however, two important challenges in the model-solver coupling remain under-explored, as discussed in the following two sub-sections.

1.4.2.1 Coupling Between Mismatched Time-Integration Schemes

It is typically assumed that the same time-integration scheme is used for both the solver and the RCM. However, this is often not desired in real applications. Actually, most RCMs, especially the non-intrusive ones, can work robustly with fewer sub-stages in the time-integration compared with a FOM, and can sometimes even work at larger time-steps. The robustness can be attributed to multiple reasons, such as the filtering (through processes like the POD projection and reconstruction) of small-scale physics where the numerical stability usually originates, or the use of an autoregressive model which directly works on the time series patterns and bypasses the CFL condition. Under these circumstances, The efficiency of the RCMs will be undermined if the time-integration is forced to be as strict as in the coupled solver.

Due to the obvious analogy in the domain decomposition and the block partitioning in a traditional parallel solver, it is natural to think of a direct application of the widely used implicit coupling schemes, such as the Newton-Raphson procedures (*Matthies and Steindorf*, 2002; *Fernández and Moubachir*, 2005), and the Schwarz procedures, including the additive/parallel ones known as the block Jacobi

method (*Schwarz*, 1972; *Shroff and Schreiber*, 1989), and the multiplicative ones known as the block Gauss-Seidel method (*Smith et al.*, 1996; *Grippo and Sciandrone*, 2000). Such implicit schemes are also known as strongly coupled schemes, as they require the full computation domain to operate in a strictly aligned time-integration scheme with multiple (often $O(10)$) iterative computations within each time-step, known as sub-iterations or sub-cyclings. The requirement, however, is not suitable for most RCMs, which often use a simpler time-integration scheme, e.g. the forward-Euler. Although in principle, an intrusive ROM can be designed to operate identically to a FOM computation-wise, a large number of sub-iterations in implicit schemes conflicts with the efficiency demand of a ROM, and thus, to the author's best knowledge, only used with inexpensive ROMs under certain linear assumptions in practice. *Kerfriden et al.* (2013) and *Corigliano et al.* (2015) performed iterative coupling between intrusive ROMs for sub-domains exhibiting linear dynamics, and FOMs for other sub-domains requiring nonlinear residuals. *Buffoni et al.* (2009) used an intrusive ROM with nonlinear residual terms, and the linear assumption is made by approximating the Jacobian matrix in the iterative Newton's algorithm at a fixed point, i.e. a quasi-Newton's method is used. Another way to avoid a mismatch in the time-integration schemes is to restrict the solver to also use a simple explicit scheme such as the forward-Euler or the RK schemes (*Huang et al.*, 2016; *Xu and Duraisamy*, 2017), or a non-iterative implicit scheme such as the backward-Euler (*Baiges et al.*, 2013). However, such choices are often not capable of dealing with stiff problems and thus impractical. Moreover, RCMs are shown to often be able to run at a larger time-step size than a solver (*Marley et al.*, 2015; *Xu and Duraisamy*, 2017; *Pfaff et al.*, 2020; *Xu et al.*, 2021), and the advantage has to be discarded when a strongly coupled scheme is used.

Schemes for coupling solvers/models with different time-step sizes and number of sub-iterations are commonly referred to as *staggered schemes* (*Felippa and Park*,

1980; *Piperno et al.*, 1995; *Felippa et al.*, 2001), also known as weakly coupled schemes. In conventional staggered schemes, communications only take place at the beginning of each time-step. Within each step, the received values from a partner remain as a fixed boundary condition regardless of the time-integration schemes used inside a sub-domain. When different time-step sizes are used, the coarser solution can be first advanced, then linearly interpolated at the finer steps, and used for advancing for the latter. In this case, a serial staggered scheme is required, which means that the computations of the sub-domains are performed sequentially. When no interpolation is needed, either with aligned time-step sizes, or by keeping the interface value fixed through multiple fine time-steps, a parallel staggered scheme can be used, in which the different sub-domains are advanced simultaneously. *Farhat and Lesoinne* (2000) proved that both the serial and the parallel conventional staggered schemes are first-order accurate, regardless of the schemes for the different sub-domains. The parallel scheme is shown to be more efficient at the cost of a slightly lower accuracy (*Farhat and Lesoinne*, 2000; *Gatzhammer*, 2014). Second-order accurate staggered schemes with backward difference have also been proposed (*Farhat et al.*, 2006, 2010). Comprehensive reviews of staggered schemes can be found in *Felippa et al.* (2001) and *Gatzhammer* (2014), and it can be realized that staggered schemes have been most widely studied in the simulations for fluid-structure interactions (*Bathe et al.*, 1995; *Löhner et al.*, 2007; *Dettmer and Perić*, 2013; *Bathe and Kamm*, 1999). Existing explorations on staggered model-solver coupling also inherit such heterogeneous partitions (*Erbts and Düster*, 2012; *Kafkas and Lampeas*, 2020), and it is hard to distinguish the potential error caused by the interface coupling from the intrinsic discontinuity in many variables across the fluid-structure interface. The performance of staggered model-solver coupling between homogeneous sub-domains remains to be studied.

1.4.2.2 Coupling with Non-Intrusive Models

Our discussions so far have been focused on the coupling scheme, without taking into account the fact that even with the same time-integration scheme, a RCM can behave differently from a FOM at the interface. In fact, the RCMs in all aforementioned successful applications, even the ones taking linear assumptions, are limited to intrusive ROMs, which are able to process the interface value from the coupled partner with the help of governing equations. However, such a capability is not guaranteed by the non-intrusive models, and few relevant studies can be found. In a series of papers (*Xiao et al.*, 2016a, 2017, 2019), researchers performed domain decomposition for a Radial Basis Function (RBF)-based non-intrusive ROMs, and demonstrated the possibility to couple non-intrusive ROMs in both ROM-FOM and ROM-ROM settings. However, the ROM-FOM coupling is again performed at fluid-structure interfaces, and the effect of the coupling scheme is not discussed.

Because of the similarity between a domain interface and a physical boundary *with transient values*, beneficial inspirations can be drawn from explorations on incorporating *parametric* boundary conditions into the non-intrusive models. *Swischuk et al.* (2020) and *McQuarrie et al.* (2021) used a linear transformation to account for the impact of an external forcing onto the POD coefficients in a non-intrusive ROM. *Xu and Duraisamy* (2020) used a MLP to predict the encoded variables at different inflow angles for an autoencoder. For non-projection-based deep learning models such as the PINN, a popular choice is to impose a soft constraint by training with an additional penalty term for the mismatches at the boundaries (*Raissi et al.*, 2019; *Márquez-Neila et al.*, 2017). For convolutional and graph-based deep learning models, a vanilla incorporation of boundary values can appear straight-forward: the convolutional domain or graph is designed to include the boundary region, and the input boundary values are automatically passed into the rest of the domain through convolutions or message passing. However, when multiple convolutional or message

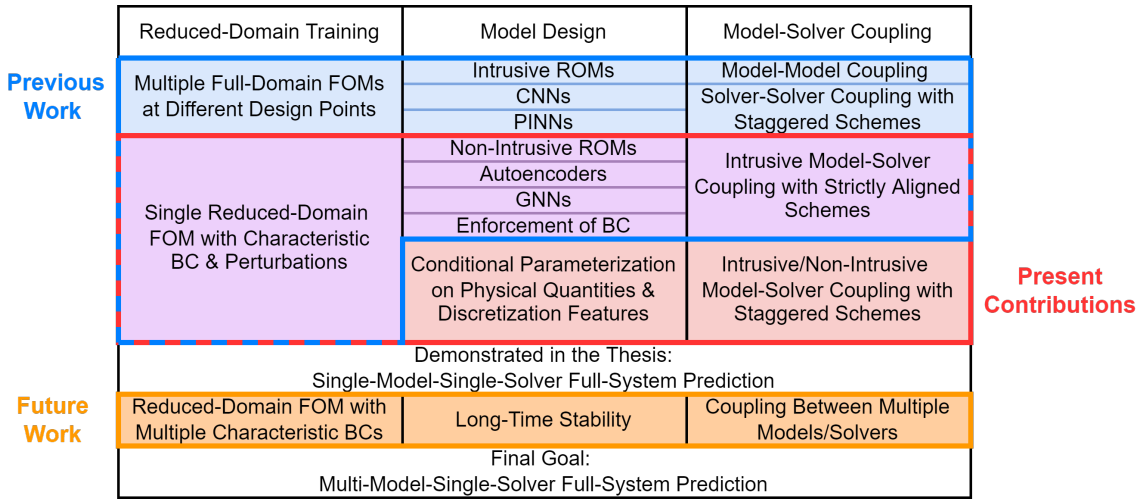


Figure 1.3: Contributions of the thesis in the roadmap towards full-system modeling.

passing layers are used, the desired size of the informative boundary region grows linearly, otherwise, a "zero" boundary condition will be implicitly imposed due to the use of padding or a missing graph edge. Additional treatments in the boundary regions are designed to alleviate the problem, such as adding labels to distinguish the boundary and inner regions (*Pfaff et al.*, 2020), or using independently trained network parameters for edges at the boundaries (*Xu et al.*, 2021). It should be mentioned that the treatments can also be used to learn non-parametric boundary conditions, such as a no-slip wall, which by itself forms a hot area of study (*Gao et al.*, 2020; *Sun et al.*, 2020; *Chen et al.*, 2021). However, such boundaries are less relevant to the sub-domain coupling of our interest, and fall outside the scope of this section.

1.5 Contributions

This thesis addresses the aforementioned challenges imposed by the low data limit in the development of RCMs. Fig. 1.3 illustrates the areas that the thesis contributes to in a roadmap towards the full-system modeling strategy introduced in Sec 1.4.1. The key contributions are listed below:

1. Multiple classes of deep-learning models are introduced to the development of

non-intrusive ROMs. Improvements over the traditional POD-based intrusive ROM are demonstrated in a number of numerical tests.

2. A model based on multi-level convolutional autoencoder networks is proposed for highly efficient parametric and future-state predictions of spatio-temporal dynamics.

3. The idea of Conditional Parameterization (CP) is generalized to enable the efficient learning of complex physical terms and mesh discretization characteristics. It is demonstrated that a drop-in CP modification can bring significant improvements for various existing models on several tasks essential to the modeling of physical systems.

4. A Conditionally Parameterized Graph Neural Network (CP-GNet) is proposed, which effectively models complex physics such as chemical source terms, irregular mesh discretizations, and different types of boundary conditions. Successful future-state predictions are performed on reacting flows, changing geometries, and cases originating from different spatial discretization methods (FEM/FVM).

5. Multiple model-solver coupling formulations are proposed and investigated. Effective incorporation of boundary/interface conditions for the non-intrusive ROMs is enabled. The staggered schemes from fluid-structure interactions are migrated for model-solver coupling with specialized improvements for individual time-integration schemes with largely mismatched numbers of sub-stages and time-step sizes.

6. The reduced-domain training, full-domain prediction framework is thoroughly investigated in a quasi-1D model at multiple operating conditions and prediction geometries.

7. The aforementioned techniques are applied jointly to extend the framework for non-intrusive models, which is verified on a realistic single-element combustor.

Most of the above contributions are documented in the following publications:

1. Xu, J., and K. Duraisamy (2017), Reduced-order modeling of model rocket combustors, in *53rd AIAA/SAE/ASEE Joint Propulsion Conference*, p. 4918.

2. Huang, C., J. Xu, K. Duraisamy, and C. Merkle (2018), Exploration of reduced-order models for rocket combustion applications, in *2018 AIAA Aerospace Sciences Meeting*, p. 1183.

3. Xu, J., C. Huang, and K. Duraisamy (2018), Multi-domain reduced-order modeling with sparse acceleration of combustion instability, in *2018 Joint Propulsion Conference*, p. 4680.

4. Xu, J., C. Huang, and K. Duraisamy (2019), Reduced-order modeling framework for combustor instabilities using truncated domain training, *AIAA Journal*, pp. 1–15.

5. Sharma, A., J. Xu, A. K. Padthe, P. P. Friedmann, and K. Duraisamy (2019), Simulation of maritime helicopter dynamics during approach to landing with time-accurate wind-over-deck, in *AIAA Scitech 2019 Forum*, p. 0861.

6. Xu, J., and K. Duraisamy (2020), Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics, *Computer Methods in Applied Mechanics and Engineering*, *372*, 113,379.

7. Xu, J., A. Pradhan, and K. Duraisamy (2021), Conditionally parameterized, discretization-aware neural networks for mesh-based modeling of physical systems, *Advances in Neural Information Processing Systems*, *34*.

1.6 Outline

The rest of the thesis is organized as follows: Chapter II demonstrates the full-system coupling framework in a simplified setting, which sets the stage for the subsequent chapters. Chapter III presents the projection-based non-intrusive ROMs. Chapter IV describes the graph representation of discretized systems and the conditional parameterization method, which is followed by the proposal of the Conditionally Parameterized Graph Neural Network (CP-GNet) model. Chapter V presents model-solver coupling methods. Besides the individual demonstrations in their corresponding chapters, the different models are integrated and demonstrated in two numerical tests at the end of Chapter V. Concluding remarks and perspectives are given in Chapter VI.

CHAPTER II

Reduced-Domain Training and Coupled Prediction

2.1 Introduction

The goal of this chapter is to provide a brief yet complete tour of the full-system modeling framework introduced in Sec. 1.4.1. For illustrative purposes, a quasi-1D representation of a model rocket combustor is addressed with a basic RCM, the static-basis intrusive ROM, which is coupled with a FOM using strictly aligned explicit time-integration schemes in the online prediction. After a clear outline of the workflow is depicted, detailed studies on more complex models, coupling schemes, and test cases are presented in the subsequent chapters.

2.2 Problem Statement

A numerical experiment is designed to demonstrate the workflow and capability of the framework in a combustion system with a variable geometry. **The task is to develop a single ROM for the reactive region only, and couple it with a FOM solver to perform predictions for multiple unseen geometries.** The study is based on a quasi-1D version of the Continuously Variable Resonance Combustor (CVRC) (*Yu*, 2009; *Yu et al.*, 2012), shown in Fig. 2.1, which is a small-scale single-injector rocket engine combustor with a variable chamber length. At differ-

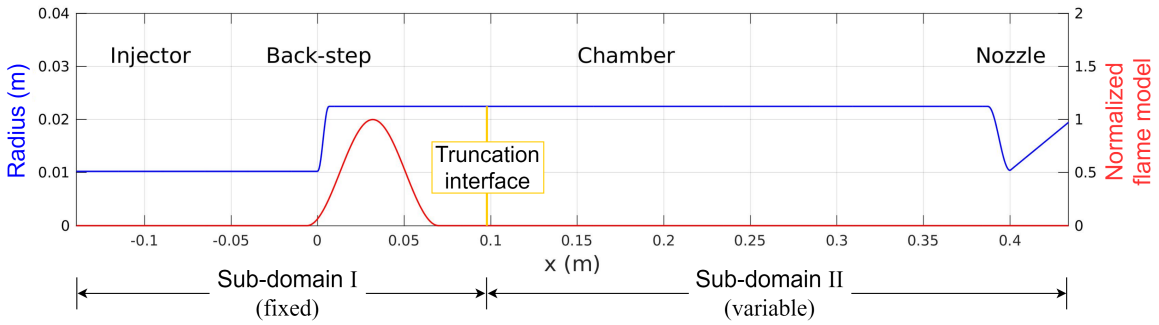


Figure 2.1: Decomposed computation domain and shape of the flame model.

ent lengths, the acoustic profile in the chamber is changed, which will consequently change the coupled combustion process, and challenges the predictive capability of the ROM.

As shown in Fig. 2.1, the domain is decomposed into two sub-domains. The truncation interface is located at $x = 0.096$ m, such that the first sub-domain contains the physics-complex areas including the injector, back-step, and the leading part of the chamber where the flame is located. The second sub-domain includes the rest of the geometry, which has a variable length in design evaluations. In our framework, a ROM is used in the first sub-domain, and a FOM is used in the second. The allocations are based on consistent assumptions with these made in Sec. 1.4.1. More specifically, for the single-element configuration used in this quasi-1D study, and its 2D high-fidelity version to be present in Sec. 5.6:

1. The complex, computation-intensive area is fully covered in the first sub-domain. The accurate modeling of this domain requires high-resolution simulations. The FOM computation in the second sub-domain is expected to be much less challenging than the first domain and affordable with coarser resolution modeling approaches (e.g. coarse-mesh LES and unsteady RANS), which makes a ROM replacement unnecessary.

2. In design evaluations, the chamber length in the second sub-domain is vari-

able. Traditionally, a new ROM training is required for each chamber length, which violates the goal of reducing the computing cost using ROM. It improves the overall efficiency of the framework to use a FOM to handle the different geometries flexibly.

2.3 Full Order Model

The quasi-1D unsteady Euler equations with species transport are taken as the FOM in this study, represented as:

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = \mathbf{S}_A + \mathbf{S}_f + \mathbf{S}_q, \tag{2.1}$$

where

$$\mathbf{q} = \begin{pmatrix} \rho A \\ \rho u A \\ \rho E A \\ \rho Y_{ox} A \end{pmatrix}, \mathbf{F} = \begin{pmatrix} \rho u A \\ \left(\rho u^2 + p\right) A \\ \left(\rho E + p\right) u A \\ \rho u Y_{ox} A \end{pmatrix}, \mathbf{S}_A = \begin{pmatrix} 0 \\ p\frac{dA}{dx} \\ 0 \\ 0 \end{pmatrix}, \mathbf{S}_f = \begin{pmatrix} \dot{\omega}_f \\ \dot{\omega}_f u \\ \dot{\omega}_f \Delta h_0 \\ -\dot{\omega}_{ox} \end{pmatrix}, \mathbf{S}_q = \begin{pmatrix} 0 \\ 0 \\ q' \\ 0 \end{pmatrix}. \tag{2.2}$$

In the conservative variable vector $\mathbf{q}$, ρ is the density, u is the velocity, E is the total internal energy, Y_{ox} is the oxidizer mass fraction, and A is the cross-sectional area. The corresponding convective fluxes are represented by the vector $\mathbf{F}$, where p is the static pressure. The first source term $\mathbf{S}_A$ accounts for the force of pressure on the area variation.

The latter two terms model the unsteady combustion process jointly. In the original experiment, the fuel is injected through an annular ring located at the back-step, and reacts at a finite rate with the oxidizer injected. In this work, we follow the choice of *Frezzotti et al.* (2015a,b) and take an infinitely-fast one-step combustion model. An important assumption behind the model is that, when fuel is injected, it will react with the oxidizer instantaneously to form products and no intermediate

species are produced, thus only one species transport equation is needed. The fuel injection and immediate consumption of oxidizer are modeled by $\mathbf{S}_f$. As suggested in the same work, to represent a realistic flame region of a finite width, and to avoid discontinuities with the infinitely fast model, the fuel is injected at a sinusoidal spatial function $\dot{\omega}_f(x)$ given by:

$$\begin{aligned} \dot{\omega}_f(x) &= \frac{\dot{m}_f}{\int (1+\sin\xi(x))dx}\left(1+\sin\xi(x)\right), \\ \xi(x) &= -\frac{\pi}{2} + 2\pi\frac{x-l_s}{l_f-l_s}, \text{with } l_s < x < l_f, \end{aligned} \tag{2.3}$$

where $\dot{m}_f$ is the total mass flow rate of fuel injection, and l_s and l_f are the starting and ending location of the flame, respectively. The resulting shape of the flame model is shown in Fig. 2.1 along with the model geometry. Due to the infinitely-fast model, the consumption rate of the oxidizer is given by a simple conversion:

$$\dot{\omega}_{ox} = \frac{\dot{\omega}_f}{C_{f/o}}, \tag{2.4}$$

where $C_{f/o}$ is the stoichiometric fuel-to-oxidizer ratio.

The last source term $\mathbf{S}_q$ models the coupling between heat release and acoustics. Introduced by *Crocco et al.* (1958), the model assumes the unsteady part of heat release as a time-delayed function of pressure, with an amplification factor α and a time lag τ, yielding:

$$q' = \dot{\omega}_f \Delta h_0 \alpha \frac{p\left(x, t-\tau\right) - \bar{p}\left(x\right)}{\bar{p}\left(x\right)}. \tag{2.5}$$

The same geometry parameters, gas properties, and operating conditions as in the work *Xu and Duraisamy* (2017) are adopted and listed in Table 2.1 and 2.2.

Table 2.1: Geometry parameters

Section	Injector	Back-step	Nozzle converging part	Nozzle diverging part
Length (m)	0.1397	0.0064	0.0127	0.034
Radius (m)	0.0102	0.0102 to 0.0225	0.0225 to 0.0104	0.0104 to 0.0195

Table 2.2: CVRC operating conditions

Parameter	Value
Fuel mass flow rate, kg/s	0.027
Fuel temperature, K	300
Oxidizer mass composition	57.6% H_2O + 42.4% O_2
Oxidizer mass flow rate, kg/s	0.32
Oxidizer temperature, K	1030
Fuel composition	100% CH_4
Equivalence ratio	0.8

2.4 Framework Details

In this section, we describe the detailed procedures for applying the multi-fidelity framework to the aforementioned domain-decomposed single-element combustor, and contrast it with conventional ROM development. A schematic is given in Fig. 2.2. The major steps of our framework include:

1. Perform a FOM training simulation on the first sub-domain with a broadband perturbation added on the truncation interface, which is treated as a characteristic boundary.

2. Generate the POD bases for different variables using SVD of the solution from the training simulation.

3. Use the POD bases in a ROM solver for the first sub-domain and couple it with a FOM solver for the variable-length, acoustics-dominated chamber for predictions.

2.4.1 Characteristic ROM Training on a Reduced Domain

For the proposed training method, a FOM simulation is performed in the first sub-domain for each α. To obtain a basis that is representative of the physics in such a reduced domain, we take a similar training procedure as in the work *Huang et al.* (2016, 2017). In this procedure, the truncation interface is treated as a characteristic

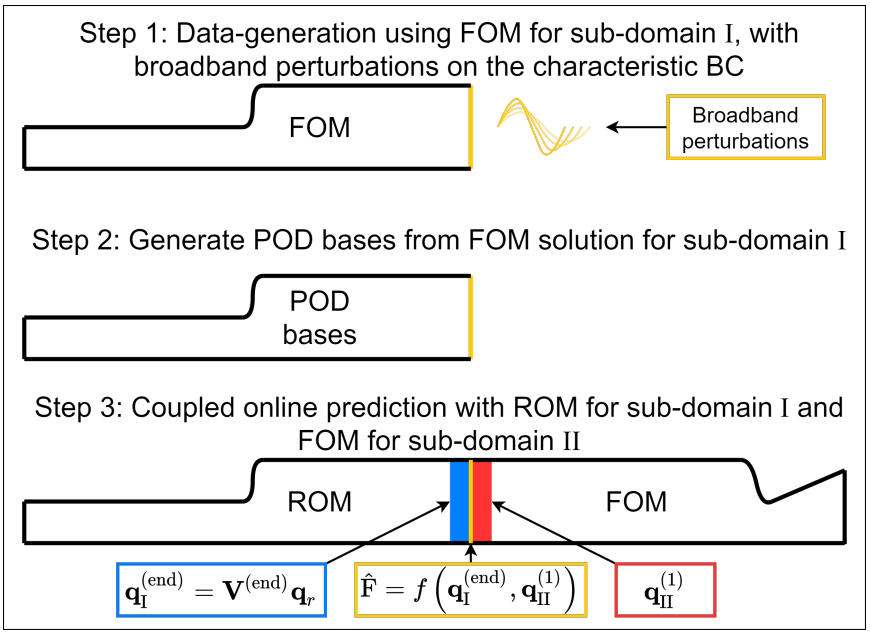


Figure 2.2: Schematic of the framework.

outlet boundary in the training FOM simulations. The specified properties at the characteristic boundary are as follows:

$$\Omega_{bc} = \{J, u, T, Y_{ox}\}^T, \tag{2.6}$$

where c is the speed of sound, $J = -\frac{p}{\overline{\rho c}} + u$ is one-dimensional approximation of the characteristic invariant for the in-coming acoustic wave, $\overline{\rho c}$ is obtained from the steady state solution, and the other three primitive variables are extrapolated from the values of the interior cells. Compared with a standard outflow condition, the characteristic boundary helps to eliminate the undesirable resonant acoustic modes corresponding to the reduced-domain geometry. As demonstrated in *Huang et al.* (2017), the presence of such resonant acoustic modes can significantly affect the predictive capabilities of the ROMs.

The FOM for the reduced-domain is also started from the steady state as in the self-excited simulations on the full domain. The difference is that instead of a single

frequency inlet perturbation, a broadband perturbation J' is imposed on the incoming (upstream-running) characteristic wave at the truncation boundary:

$$J'(t) = 0.01 J_0 \sum_{i=1}^{n_f} \sin\left(2\pi(f_0 + (i-1)\Delta f)t\right), \tag{2.7}$$

where the range of frequencies specified by f_0, Δf, and n_f can be estimated from *a priori* analysis (*Grenda et al.*, 1995) such that the range of resonant frequencies corresponding to different chamber lengths are covered. This perturbation is imposed over the entire duration of the FOM simulation. The bases $\mathbf{V}$ for the conserved variables are then obtained from the reduced domain solution snapshots, and used in POD-Galerkin projection to derive the ROM for the first sub-domain.

2.4.2 ROM-FOM Coupling in Online Prediction

In the online predictions for each α, the same ROM is used, in coupling with FOMs for different chamber lengths. In this chapter, we consider strictly aligned time-integration schemes – both models are advanced with forward-Euler at the same time-step size. In this case, the sub-domains communicate at every time-step by exchanging the cell-values adjacent to the interface. Similar to the interior of the computing domain, Roe's upwind flux *Roe* (1986) is used at the interface. Using subscript I and II for the first and second sub-domain, and superscript (1) and (end) for the first and last cell in the corresponding sub-domain, respectively, the flux at the interface is given by

$$\hat{\mathbf{F}} = \frac{1}{2}\left(\mathbf{F}_{\mathrm{I}}^{(\mathrm{end})} + \mathbf{F}_{\mathrm{II}}^{(1)}\right) - \frac{1}{2}\left|\frac{\partial \mathbf{F}}{\partial \mathbf{q}}\left(\mathbf{q}^*\right)\right|\left(\mathbf{q}_{\mathrm{II}}^{(1)} - \mathbf{q}_{\mathrm{I}}^{(\mathrm{end})}\right), \tag{2.8}$$

where $\mathbf{q}^*$ is the Roe-averaged state calculated from $\mathbf{q}_{\mathrm{I}}^{(\mathrm{end})}$ and $\mathbf{q}_{\mathrm{II}}^{(1)}$. $\mathbf{q}_{\mathrm{I}}^{(\mathrm{end})}$ is computed from the last row of the basis and the reduced variable, $\mathbf{q}_{\mathrm{I}}^{(\mathrm{end})} = \mathbf{V}^{(\mathrm{end})}\mathbf{q}_r$. The interface flux represents the necessary information for the coupling between the sub-

domains, and is used as part of the nonlinear term $\mathbf{f}$ in the corresponding ROM Eq. (1.6) and FOM Eq. (1.2), respectively.

2.4.3 Control Groups

The methodology detailed above is compared to the following control groups using the conventional ROM approach:

Control group A also uses a hybrid multi-domain solver, i.e. the first sub-domain is solved using the ROM and the second solved using the FOM. The difference from the proposed framework is in the training data generation and collection stage. In control group A, for each combination of chamber length and α, a full-domain FOM simulation is performed instead of the proposed characteristic training on the truncated domain. Then the solution is restricted to the first sub-domain and collected to generate the POD bases for the ROM of the sub-domain.

Control group B uses a traditional full-domain ROM. The same FOM training simulations as in control group A are used and the POD bases are directly generated on the full-domain solution.

To summarize, let n_{L_c} and n_α be the number of chamber lengths and amplification factors studied, respectively. In the proposed framework, n_α FOMs simulated on the reduced domain are used in total [1] using characteristic perturbations. Then the first sub-domain is simulated using the ROM, second sub-domain using the FOM. In control groups A&B, $n_{L_c} \times n_\alpha$ self-excited FOM simulations are conducted. In group A, the first sub-domain is solved using a ROM, and the second is solved using the FOM. In group B, the whole domain is solved using a ROM.

For conciseness, the proposed framework will be referred to as "Reduced-Domain training and Multi-Domain prediction" (RD-MD), control group A as "Full-Domain

[1] although as shown in the next section, this can be reduced

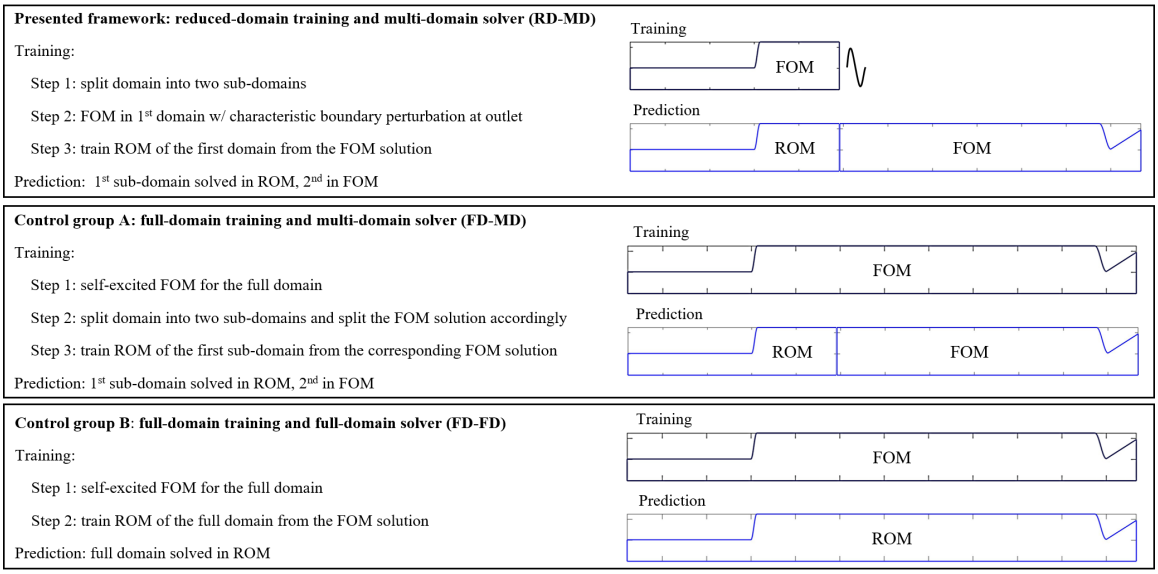


Figure 2.3: Comparison of training (outlined in black) and prediction (outlined in blue) approaches.

training and Multi-Domain prediction" (FD-MD), and control group B as "Full-Domain training and Full-Domain prediction" (FD-FD). A schematic of the different methods is given in Fig. 2.3.

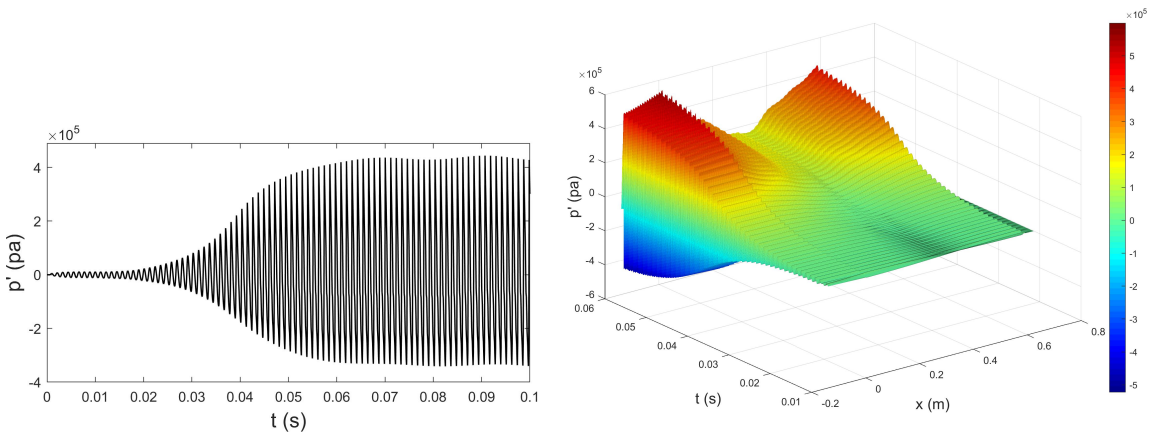
2.5 FOM Results

Numerical tests are conducted for chamber lengths L_c ranging from 0.254 m to 0.762 m at an interval of $\Delta L_c = 0.0635$ m, each at three amplification factors $\alpha = 3.1, 3.25$ and 3.4. For each test, a steady state is first achieved with the unsteady source term $\mathbf{S}_q$ turned off. From the steady state, a low-amplitude perturbation is applied to the inlet boundary to trigger the instability at the beginning of predictive unsteady simulations (either FOM or ROM). At different combinations of L_c and α, two general categories of responses after the termination of perturbation are identifiable: one with positive growth rate, in which the pressure oscillation grows and settles into a limit cycle oscillation (LCO), and one with negative growth rate in which the instability starts to decay after the perturbation ends and the flow con-

verges to a steady state again. The predicted response is visualized using pressure signals obtained 0.0127 meters upstream of the converging part of the nozzle, which is a typical location selected to probe combustion instabilities (*Frezzotti et al.*, 2015b). The definition of the growth rate, gr, is based on the peak-to-peak amplitude of the unsteady part of the pressure signal for $t = [0.01, 0.05]$ s, the growth for which is fitted to an exponential function:

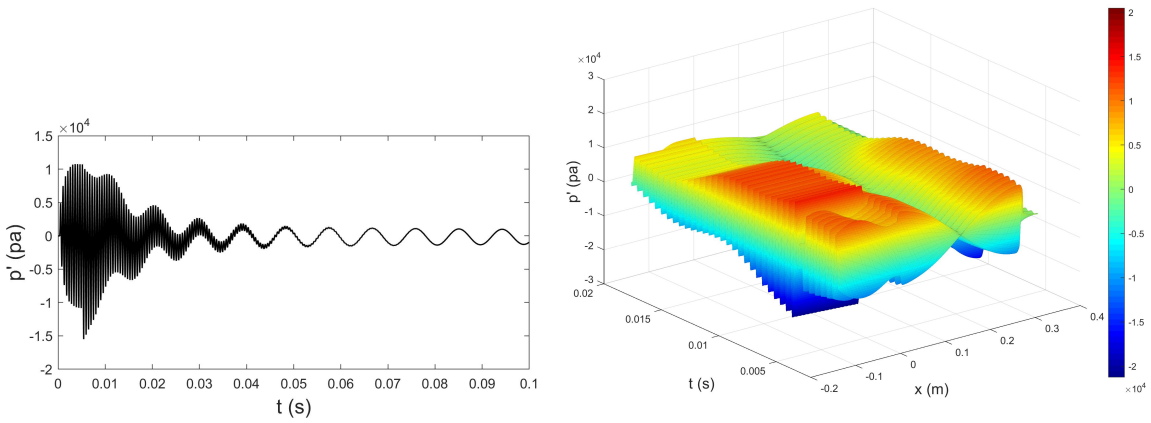
$$|p'(t)| = |p'(0.01)|e^{(t-0.01)\cdot gr}. \tag{2.9}$$

To better distinguish the categories, two representative examples (at $\alpha = 3.4, L_c = 0.5715$ m for the growing category and $\alpha = 3.1, L_c = 0.254$ m for the decaying category, respectively) are presented in Fig. 2.4 and 2.5, where the difference in growth rate can be clearly observed via the monitored pressure signals and the spatio-temporal diagrams of the pressure evolution. Similar responses can be found in previous studies *Xu and Duraisamy* (2017); *Wang et al.* (2019a).



(a) Pressure signal at the monitored location (b) Spatio-temporal diagram of the pressure evolution

Figure 2.4: Example response with positive growth rate, $\alpha = 3.4, L_c = 0.5715$ m.



(a) Pressure signal at the monitored location (b) Spatio-temporal diagram of the pressure evolution

Figure 2.5: Example response with negative growth rate, $\alpha = 3.1, L_c = 0.254$ m.

2.6 ROM Results

From *a priori* analysis (*Grenda et al.*, 1995), the longitudinal frequency of the chamber spans approximately between 700 and 2000 Hz for the tested configurations, and the parameters for broadband perturbation signal Eq. (2.7) is set to $f_0 = 700$ Hz, $\Delta f = 100$ Hz, $n_f = 14$.

To train the control groups, a full-domain self-excited FOM simulation is conducted at each combination of L_c and α. In all three methods, snapshots of the training solution are collected every 100 time-steps over a period of 0.5 s following the initial steady state.

2.6.1 Singular Values and Offline Projection-Reconstruction

The POD singular values from different training methods are compared at $\alpha = 3.25, L_c = 0.254, 0.508, 0.762$ m. Since the number of snapshots is larger than the mesh size, the maximum number of modes and singular values from SVD is limited to the mesh size, n_x. Fig. 2.6 shows the complementary part of the normalized cumulative sum η of the singular values at $\alpha = 3.25$. For the first n_r POD modes, η

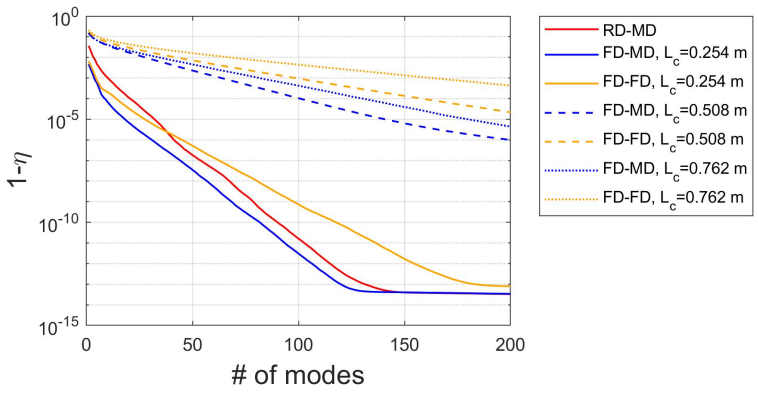


Figure 2.6: The complementary part of cumulative energy in the singular values at $\alpha = 3.25$

is given by:

$$\eta_{n_r} = \sum_{i=1}^{n_r} \sigma_i / \sum_{i=1}^{n_x} \sigma_i. \tag{2.10}$$

It should be noted that there is only one curve for RD-MD as it uses the same training simulation and POD bases for all chamber lengths. It can be observed that for both FD-MD and FD-FD, the decay in the complementary part increases as the chamber length decreases. This is due to the fact that the dynamics have a higher frequency when the chamber length is shorter, which is easier to be captured by fewer modes in SVD. In all cases, the decay in FD-FD is slower than FD-MD, because it covers a larger domain with more spatial variations in geometry and physics, and more modes are therefore needed to contain the same portion of information stored in the training data. For medium-to-high L_c, the proposed framework gives the best decay since its training data contains more high-frequency contents. At small L_c, the decay in the leading modes for the two full-domain training methods becomes slightly better than RD-MD as the response frequency in their training data is comparable to the highest one included in the reduced-domain training, whereas there are also lower frequencies contained in the latter. Results for other variables and α follow a similar pattern.

More intuitively, the quality of a set of projection bases can be evaluated by

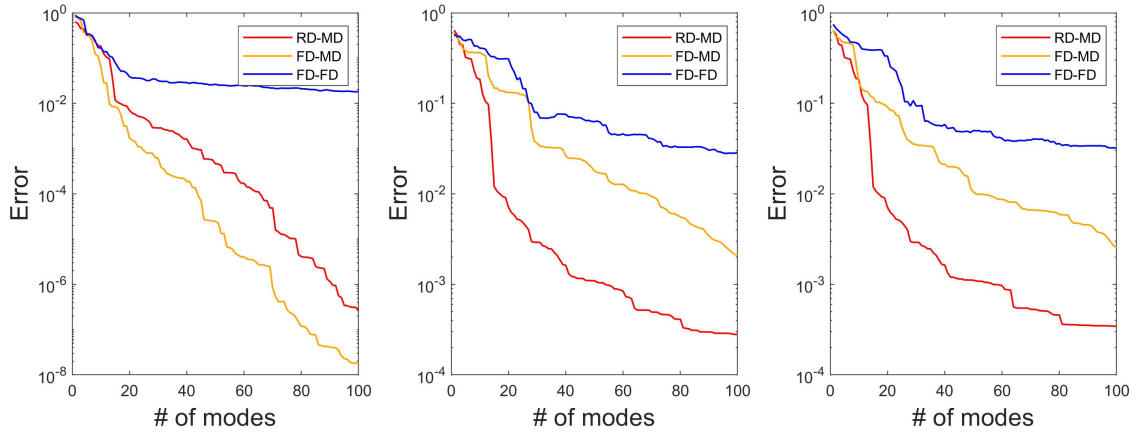
performing projection-reconstruction with it. For a given set of POD bases $\mathbf{V}$, the approximation $\tilde{\mathbf{Q}}$ to the snapshot matrix $\mathbf{Q}$ using POD projection-reconstruction is given by:

$$\tilde{\mathbf{Q}} = \mathbf{V}\mathbf{V}^T\mathbf{Q}. \tag{2.11}$$

The accuracy of the approximation can be evaluated using the relative L2 error, defined as:

$$\epsilon = \frac{\left\|\mathbf{Q} - \tilde{\mathbf{Q}}\right\|_2}{\|\mathbf{Q}\|_2}. \tag{2.12}$$

Again, at $\alpha = 3.25$, $L_c = 0.254, 0.508, 0.762$ m, error in the projection-reconstruction results at different sizes of bases up to $n_r = 100$ is shown in Fig. 2.7. It can be observed that both RD-MD and FD-MD provide a monotonic decrease in the projection error. However, for FD-FD, the error increases with n_r in a few cases, which is not desirable for ROM development as it makes the choice of basis size more uncertain. The increase can be again attributed to the fact that the POD method solves a global minimization problem and the optimal bases for the full domain are not necessarily optimal for the reduced domain. The improved projection error property with the multi-domain method implies a higher accuracy and stability of ROM, which is confirmed in the following sections.



(a) $L_c = 0.254$ m (b) $L_c = 0.508$ m (c) $L_c = 0.762$ m

Figure 2.7: Relative L2 error in POD projection-reconstruction for $\alpha = 3.25$.

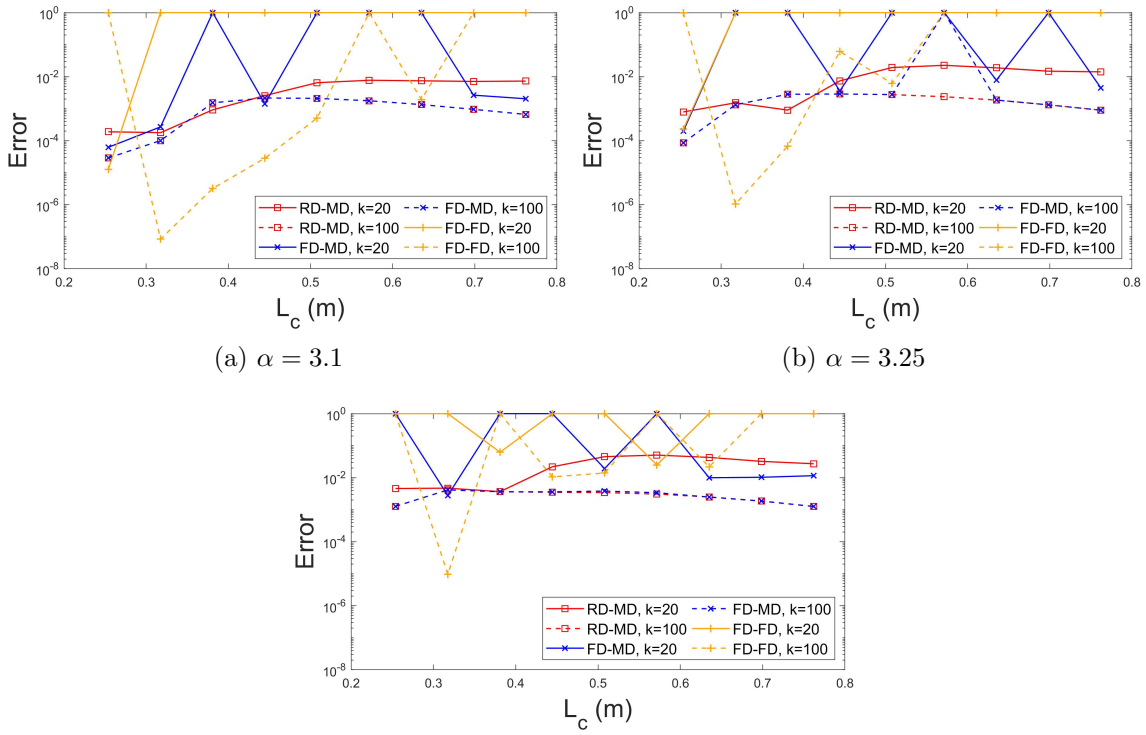
2.6.2 Coupled Online Prediction

The three methods are compared at the two basis sizes, $n_r = 20, 100$. The relative L2 error of the ROM solution is shown in Fig. 2.8. It should be mentioned that in several cases, the ROMs in the control groups are numerically unstable. The error is set to 1 under these circumstances, which is higher than the error in all stable cases.

When the basis size is sufficiently large, i.e. at $n_r = 100$, RD-MD performs similarly to FD-MD, except for one case at $\alpha = 3.25, L_c = 0.5715$ m, where the latter is unstable. At high α, where the instability is stronger, the advantage of RD-MD becomes more significant. This can be observed from the medium L_c cases at $\alpha = 3.4$, where the gap between RD-MD and FD-MD is apparent. Also, FD-FD performs worse as α increases, resulting in higher errors and more unstable cases. It should be noted that FD-FD sometimes outperforms the other two methods at low L_c, which is consistent with an improved decay in the singular values. However at $L_c = 0.254$ m, it becomes unstable.

When the basis size is reduced to $n_r = 20$, all the predictions deteriorate considerably. The two full-domain training methods become unstable in most conditions. In contrast, RD-MD remains stable at all combinations of parameters, although the ROM solution error is approximately one order higher than that of the $n_r = 100$ cases. The advantage in stability confirms the conclusion that the reduced-domain training results in a better set of basis functions. By virtue of using a smaller basis set, the computational cost, e.g. the projection computation in Eq. (1.6), decreases linearly when an explicit time-marching scheme is used. In contrast to the traditional full-domain training methods, for which the stability is less predictable, the proposed framework provides a broader stability envelope, and consequentially a more flexible balance between ROM accuracy and efficiency.

Major quantities of interest in rocket combustor design include the dominant acoustic frequencies, the growth rates, and LCO peak-to-peak amplitudes of the pres-



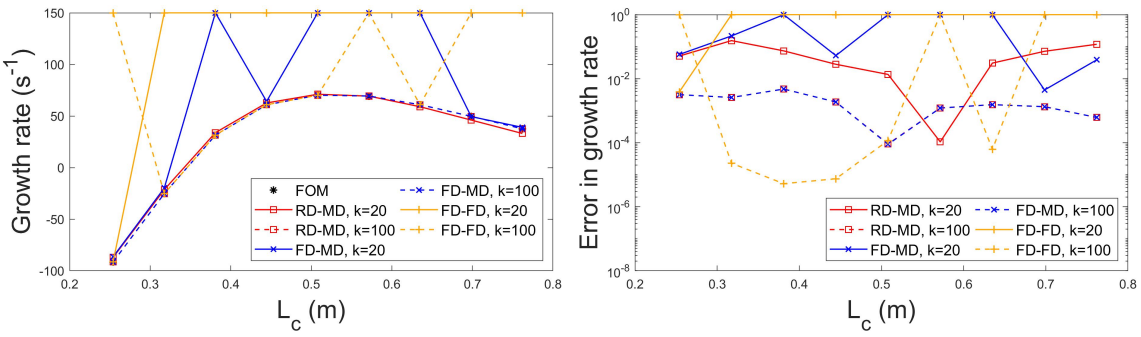
(a) $\alpha = 3.1$ (b) $\alpha = 3.25$

(c) $\alpha = 3.4$

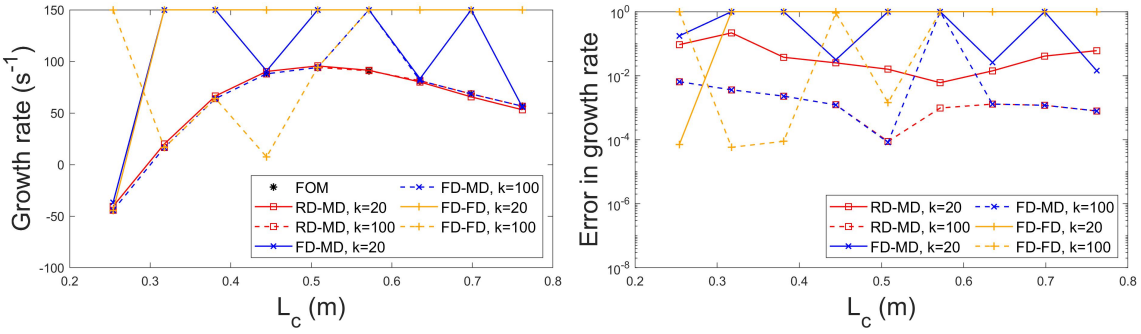
Figure 2.8: Relative L2 error in ROM solutions. Numerically unstable cases are shown as ceiling points.

sure oscillations. It should be noted that the dominant acoustic frequencies have been well-predicted by all three methods. Therefore only the comparisons of the other two quantities, growth rates and LCO amplitudes, are shown in the current study, which are visualized in Figs. 2.9 and 2.10, respectively. The relative performance between different methods follows a similar relation as in the L2 error analysis. It can be seen that the proposed RD-MD framework is able to predict the relation between the growth rate and L_c accurately with an error below 0.5% at $n_r = 100$ and below 5% at $n_r = 20$ in most cases, which illustrates its effectiveness.

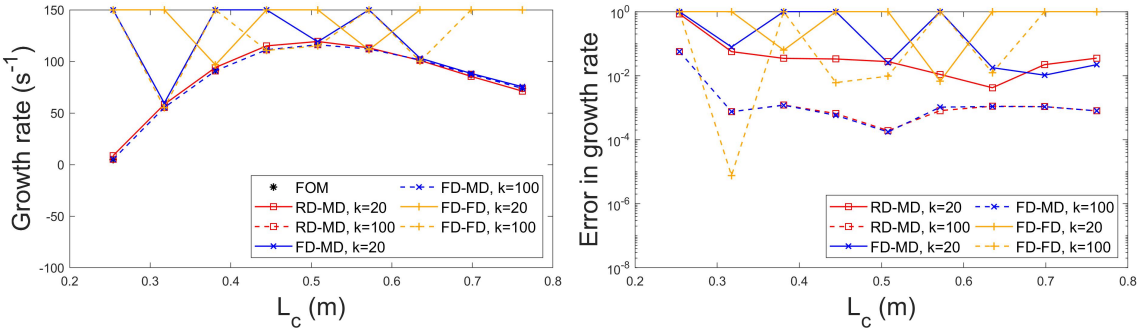
Finally, to provide a more direct comparison between the ROM solution from the framework and the FOM solution, spatial pressure profiles for $L_c = 0.508$ m, $\alpha = 3.25, n_r = 100$ are provided at several time instances in Fig. 2.11. These plots are focused on the back-step area for better visualization, and are selected by the



(a) $\alpha = 3.1$



(b) $\alpha = 3.25$



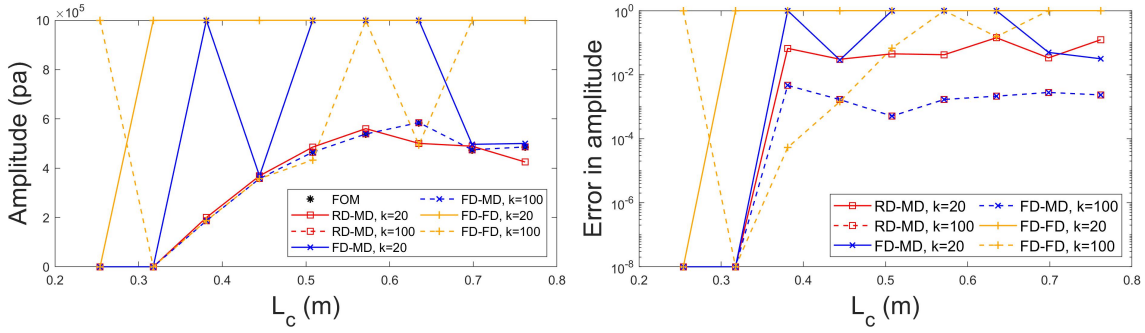
(c) $\alpha = 3.4$

Figure 2.9: Predicted growth rate and error.

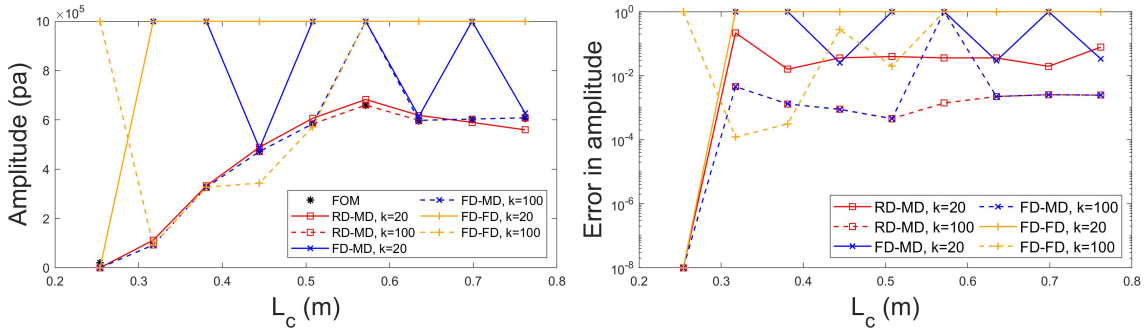
end of the simulation when the error is maximum. It is observed that the RD-MD solution correlates well with the FOM, and connects smoothly across the interface.

2.6.3 Off-Design Condition Performance

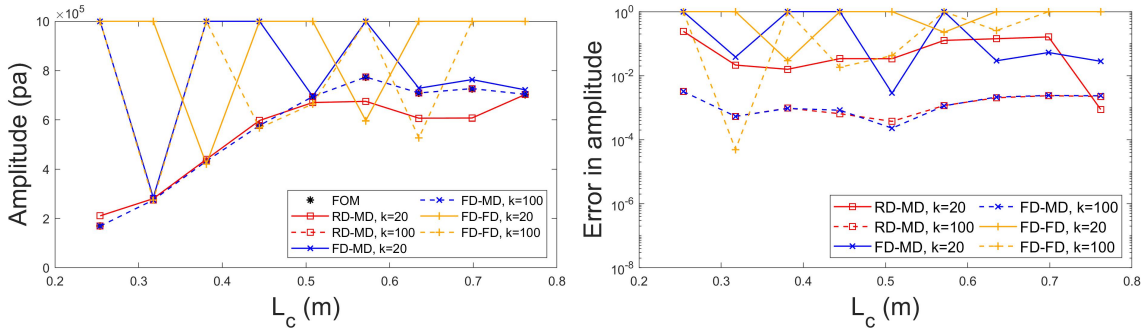
To further assess the RD-MD framework, the ROM trained using $\alpha = 3.25, n_r = 100$ is evaluated at several off-design conditions. The evaluations include two cases at chamber lengths $L_c = 0.1905$ m and 1.016 m. These cases are characterized



(a) $\alpha = 3.1$



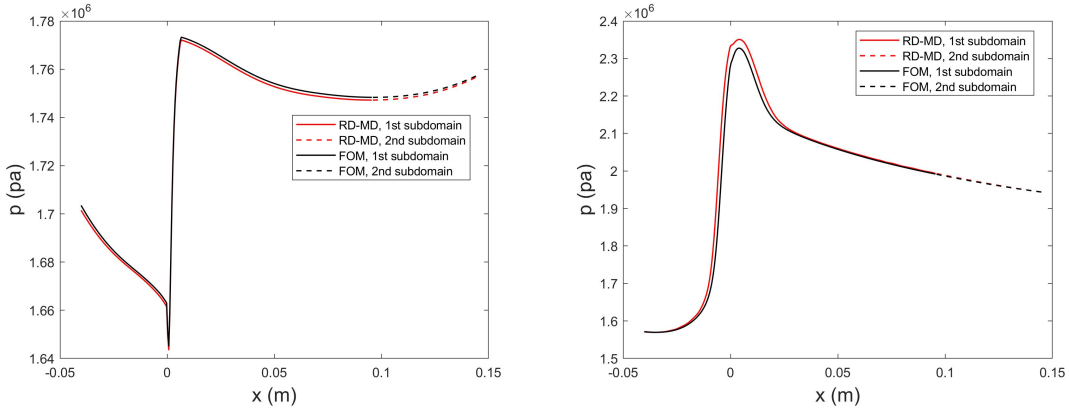
(b) $\alpha = 3.25$



(c) $\alpha = 3.4$

Figure 2.10: LCO peak-to-peak amplitude of pressure oscillation 0.0127 m upstream of the nozzle

by dominant acoustic frequencies 2550 and 500 Hz, respectively, which are outside the range of the training frequencies given by Eq. (2.7). The other conditions have amplification factors that deviate significantly from the training simulation, including $\alpha = 2.85, 3.05, 3.45, 3.65$. The results for the relative L2 error in ROM solutions (ϵ_q), frequency (f), growth rate (gr), LCO peak-to-peak amplitude (amp), and their relative errors are summarized in Table 2.3. It should be noted that due to the



(a) $t = 0.098$ s (b) $t = 0.09825$ s

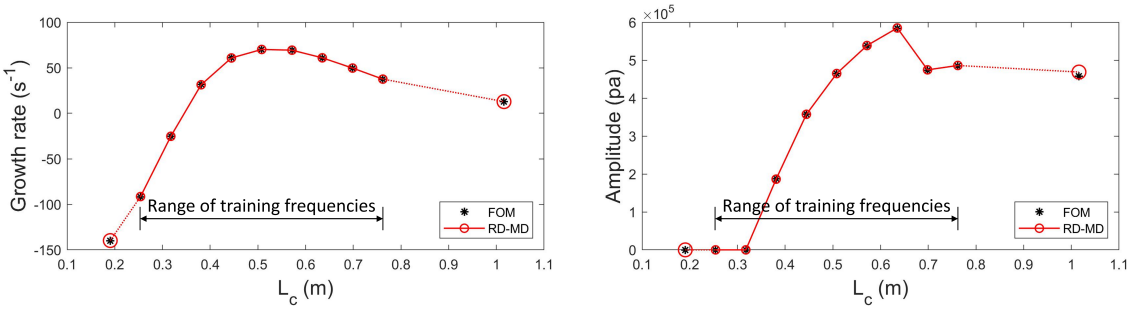
Figure 2.11: Spatial profiles of pressure

characteristic training method, the training case does not have a growing response as in the predictions, thus no growth rate or LCO amplitude is reported for the training set. Moreover, the results are plotted along with the designed conditions in Fig. 2.12 and 2.13 for a better illustration of their relation.

It is observed that, at all the off-design conditions, the relative L2 errors are below 1.2×10^{-3}. For the case with shorter L_c and higher instability frequency (OD L_{c1}), and the four cases with deviated α (OD α_1 to OD α_4), the ROM performance is comparable to that in the designed conditions. However for the case with longer L_c and lower instability frequency (OD L_{c2}), the error in LCO amplitude is 2.5%, which is more than 5 times higher than the other cases. The result indicates that for this problem, when operating within the frequency range of the training perturbation, the RD-MD method is not only independent of the chamber lengths, but also insensitive to changes in the unsteady heat release term. When beyond the training range, the instability frequency influences the predictive capabilities of the framework, and demonstrates the importance of the multi-frequency perturbation in the characteristic training.

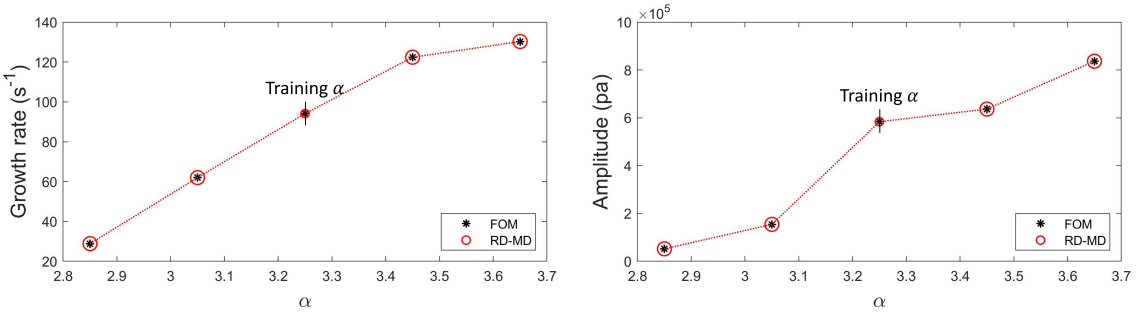
Table 2.3: Off-design condition results (f, gr, amp are listed in a "FOM/ROM" style, all errors are relative)

Case	L_c (m)	α	ϵ_q	f (Hz)	gr (s^{-1})	ϵ_{gr}	amp (Mpa)	ϵ_{amp}
Training	N/A	3.25	N/A	700 to 2100	N/A	N/A	N/A	N/A
OD L_{c1}	0.1905	3.25	9.6×10^{-5}	2516/2516	-139.98/-139.79	1.4×10^{-3}	0/0	0
OD L_{c2}	1.016	3.25	2.7×10^{-4}	490/490	12.91/12.84	5.0×10^{-3}	0.4576/0.4688	2.5×10^{-2}
OD α_1	0.508	2.85	1.2×10^{-4}	1000/1000	28.87/28.87	2.3×10^{-5}	0.0522/0.0521	1.4×10^{-3}
OD α_2	0.508	3.05	3.3×10^{-4}	1000/1000	61.99/62.00	9.1×10^{-5}	0.1545/0.1543	1.2×10^{-3}
OD α_3	0.508	3.45	1.2×10^{-3}	980/980	122.38/122.41	2.5×10^{-4}	0.6366/0.6366	8.5×10^{-5}
OD α_4	0.508	3.65	1.6×10^{-3}	980/980	130.19/130.27	6.3×10^{-4}	0.8368/0.8368	4.4×10^{-5}



(a) Growth rate (b) LCO peak-to-peak amplitude

Figure 2.12: Results at off-design L_c



(a) Growth rate (b) LCO peak-to-peak amplitude

Figure 2.13: Results at off-design α

2.7 Summary

In this chapter, the full-system modeling framework is investigated for POD-based intrusive ROMs on a quasi-1D problem. An $\alpha - \tau$ model that couples the pressure oscillation and heat release is used to represent the unstable behavior of the combustor. In the proposed framework, the domain is split into two sub-domains with the first sub-domain containing the main reaction dynamics and the second covering the adjustable chamber of the CVRC. Characteristic training is conducted in the first

sub-domain, resulting in a ROM that can be directly integrated with a FOM solver of the second sub-domain with any length.

Numerical tests are conducted at different chamber lengths L_c and amplification factors α. The framework is compared against the FOM solution, and other ROM approaches where traditional training approaches are taken. A major advantage of the current framework is that it significantly reduces the number of FOM simulations required to train ROMs while the traditional method requires a separate FOM simulation for each individual target chamber length, which does not fit the needs for a more efficient rocket engine design. Moreover, the proposed method shows faster decay of the singular value spectrum at medium-to-high chamber lengths, which implies better basis quality and ROM reliability at a low number of modes. The advantage is especially distinct in the tests with $n_r = 20$, where the proposed method shows significantly improved stability.

In predictive tests at conditions outside the training set, the framework showed significant improvement in both stability and accuracy over the traditional methods, especially when the instability is more pronounced at high α, and when the number of modes is low. Comparisons of L2 error, growth rate, and LCO peak-to-peak amplitude show that the framework is able to predict accurately at all combinations of α and L_c at $n_r = 100$.

In summary, with a significant reduction in the number of training runs needed, and an improvement in the predictive capability for off-design conditions, the multi-fidelity framework proves to be a promising approach for modeling rocket combustion instability. While the results are encouraging, the success is limited to intrusive ROMs and strictly aligned time-integration schemes, which is in favor of the model-solver coupling in the online stage, but can be less efficient compared with non-intrusive models at coarser time-step sizes. It should also be recognized that the flow and combustion models used in this study are highly simplified. In the following chapters,

more complex RCMs, and mismatched time-marching schemes are studied on more complex flow problems.

CHAPTER III

Non-Intrusive ROMs

3.1 Introduction

As introduced in Sec. 1.2.3, ROMs are characterized by formulating computations on a low-order space in an approximation to the original high-order system, and can be categorized into intrusive and non-intrusive types based on the use of governing equations. Intrusive ROMs, such as the POD-Galerkin ROM demonstrated in the previous chapter, benefit from the access to the equations, and exhibit advantages such as a low model complexity and straight-forward implementations of boundary conditions and inter-domain communications. However, at the same time, multiple limitations also originate from the intrusive procedure, major ones including:

1. The original source code for the FOM solver is required.

2. The computation for nonlinear terms still takes place in the high-order space. Acceleration methods such as the DEIM can help, but may not be effective in chaotic multi-scale problems.

3. A formal Galerkin/Petrov-Galerkin projection limits the high-dimensional part (such as the residual computation) of a model to the original spatial data structure. This limits the application of certain model reduction methods such as the convolutional autoencoder to be introduced in this chapter.

4. The solution process has to respect the form of a PDE/ODE, and be conducted in an iterative/"rollout" manner, despite a possibly different time-marching scheme.

Non-intrusive ROMs, in contrast, bypass the governing equations and can be viewed as completely data-driven surrogate models. In this vein, complex models such as deep neural networks have been extensively applied, and the advantages and disadvantages listed above are reverted, which, in many cases, results in a more efficient model. In this chapter, common neural network layers and architectures are introduced, which is followed by the explorations for multiple different techniques in the development of non-intrusive ROMs, including a nonlinear dimensionality reduction model – the (convolutional) autoencoder, several autoregressive models for the rollout prediction of reduced-order variables, and lastly a model based on multi-level autoencoder networks for a "one-shot" parametric prediction of spatio-temporal dynamics without any time-integration.

3.2 Nonlinear Dimensionality Reduction with Neural Networks

3.2.1 Feed-Forward Network

A neural network is a computation model that is comprised of stacked layers of "neurons". The model inputs are taken as the neurons for the first layer, i.e. the input layer. The neurons for the last layer, i.e. the output layer, are used as the output of the model. Any layer between the input and the output is called a hidden layer. The type of a layer is determined by the underlying computation, which in general is comprised of a set of trainable parameters, a linear operation, and a nonlinear activation function. Besides the different types of layers used, a neural network is also characterized by its architecture, i.e. how layers are interconnected.

The *feed-forward network* is the simplest architecture. The computation of a L-layer feed-forward network can be expressed as:

$$\mathbf{h}^{(L)} = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(2)} \circ f^{(1)}(\mathbf{h}^{(0)}; \mathbf{\Theta}^{(1)}), \tag{3.1}$$

where $\mathbf{h}^{(l)}$ denotes the output from the l-th layer, which is called a *hidden output* or a *latent variable*, and can be either a vector, a matrix or a tensor. $f : \mathbb{R}^{n_h} \to \mathbb{R}^{w}$ denotes the function form of a hidden layer. The most popular choice is the *dense* layer (also called the fully-connected layer), given by:

$$f_{\text{dense}}(\mathbf{h}; \mathbf{\Theta}) = \sigma(\mathbf{W}\mathbf{h} + \mathbf{b}), \tag{3.2}$$

where n_h is the input size, and w is the output size, which is called the *width* of the layer. The parameter $\mathbf{\Theta} \in \mathbb{R}^{w \times (n_h+1)}$ consists of a weight part $\mathbf{W} \in \mathbb{R}^{w \times n_h}$, and a bias part $\mathbf{b} \in \mathbb{R}^{w}$. σ is an element-wise *activation function*, which is usually nonlinear, e.g. the ReLU function. A feed-forward network consists of only dense layers is called a *Multi-Layer Perceptron (MLP)*.

3.2.2 Autoencoder

Eq. (3.1) is a serial process, which can be cut after any hidden layer, indexed by l, into two parts, $\mathbf{\Phi} : \mathbb{R}^{n_h^{(0)}} \to \mathbb{R}^{n_h^{(l)}}$ and $\mathbf{\Psi} : \mathbb{R}^{n_h^{(l)}} \to \mathbb{R}^{n_h^{(L)}}$ as:

$$\mathbf{h}^{(l)} = \mathbf{\Phi}(\mathbf{h}^{(0)}; \mathbf{\Theta}_\Phi) = f^{(l)} \circ f^{(l-1)} \circ \cdots \circ f^{(2)} \circ f^{(1)}(\mathbf{h}^{(0)}; \mathbf{\Theta}^{(1)}), \tag{3.3}$$

$$\mathbf{h}^{(L)} = \mathbf{\Psi}(\mathbf{h}^{(l)}; \mathbf{\Theta}_\Psi) = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(l+2)} \circ f^{(l+1)}(\mathbf{h}^{(l)}; \mathbf{\Theta}^{(l+1)}), \tag{3.4}$$

where the intermediate variable $\mathbf{h}^{(l)}$ can be viewed as a set of latent representations for the full variable $\mathbf{h}^{(L)}$.

The separated computation, Eq. (3.3) and Eq. (3.4), represents the functional form

of an *autoencoder*, which is a type of feedforward network with two main characteristics: 1) The output is a reconstruction of the input, i.e. $\mathbf{h}^{(L)} \approx \mathbf{h}^{(0)}$; 2) Autoencoders - in general - take a converging-diverging shape, i.e. the size of the latent variable first decreases then increases along the hidden layers. By cutting an autoencoder after its "bottleneck", i.e. the hidden layer with the smallest size, the leading part $\mathbf{\Phi}$ will compress $\mathbf{h}^{(0)}$ into $\mathbf{h}^{(l)}$ with the dimension reduced from $n_h^{(0)}$ to $n_h^{(l)}$, and the following part $\mathbf{\Psi}$ will try to recover the approximation $\mathbf{h}^{(L)}$ from $\mathbf{h}^{(l)}$.

In an autoencoder, $\mathbf{\Phi}$ is called an *encoder*, and the transformation to the latent space is called *encoding*. $\mathbf{\Psi}$ is referred to as a *decoder*, and the reconstruction process is called *decoding*. The latent variable $\mathbf{h}^{(l)}$ is commonly referred to as *code*, and $n_h^{(l)}$ is called latent dimension.

In the problems of our interest, $\mathbf{h}^{(0)}$ and $\mathbf{h}^{(L)}$ would be $\mathbf{q}$ and its approximation. Compared with the POD-based dimensionality reduction, encoding is analogous to the POD projection, decoding is analogous to the POD reconstruction, and the code becomes a reduced order variable for $\mathbf{q}$. Due to the analogy, an autoencoder is considered as a model for a nonlinear projection. For consistency and simplicity, we will use $\mathbf{r}$ to denote the reduced order variable regardless of the projection process in the rest of the paper.

3.2.3 Convolutional Autoencoder

Another type of layer widely used in neural networks is the *convolutional layer*. A convolutional layer convolves filters with trainable weights with the inputs. Such filters are commonly referred to as *convolutional kernels*. A network is called a CNN when one or more convolutional layers are used. On the same spatial grid point used in a CNN, the inputs and outputs can both have multiple parallel sets of variables called *channels*, such as the red, green, and blue components in a colored digital image. For a convolutional layer with n_{in} input channels and n_{out} output channels,

the total number of convolutional kernels is $n_k = n_{\text{in}} \times n_{\text{out}}$. Each kernel slides over the one input channel along all spatial directions, and one dot product is computed at each sliding stop, i.e. a spatial location.The functional form of a convolutional layer is given by:

$$f_{\text{conv}}(\mathbf{h}; \mathbf{k}) = \sigma(\mathbf{h} * \mathbf{k}). \tag{3.5}$$

In a 2D problem, at a sliding stop centered at (i_x, i_y), the 2D convolution of a kernel $\mathbf{k} \in \mathbb{R}^{(2w_x+1)\times(2w_y+1)}$ on an input $\mathbf{h}$ is given by

$$(\mathbf{h} * \mathbf{k})_{i_x,i_y} = \sum_{p=w_x}^{-w_x} \sum_{q=w_y}^{-w_y} \mathbf{h}_{i_x-p,i_y-q} \mathbf{k}_{p,q}. \tag{3.6}$$

Eq. (3.6) can be easily generalized for 1D or 3D as in Eq. (3.7) and (3.8):

$$(\mathbf{h} * \mathbf{k})_{i_x} = \sum_{p=w_x}^{-w_x} \mathbf{h}_{i_x-p} \mathbf{k}_p, \tag{3.7}$$

$$(\mathbf{h} * \mathbf{k})_{i_x,i_y,i_z} = \sum_{p=w_x}^{-w_x} \sum_{q=w_y}^{-w_y} \sum_{r=w_z}^{-w_z} \mathbf{h}_{i_x-p,i_y-q,i_z-r} \mathbf{k}_{p,q,r}. \tag{3.8}$$

An autoencoder with convolutional layers is called a Convolutional Autoencoder (CAE). To perform efficient dimensionality reduction, local pooling layers are appended to the convolutional layers in the encoder $\mathbf{\Phi}$, in which local neighbors of spatial points are down-sampled into single points through homogeneous computations such as taking the maximum (max-pooling) or average (average-pooling). The range of the neighbor is determined by a pooling kernel, usually defined by a uniform shape such as a 3×3 square.

The convolution-pooling operations can only perform dimensionality reduction in a division manner, which brings two limits: 1) the dimension (number of axes) of the input will not be changed. 2) The choice of n_r is not arbitrary. To improve the flexibility of a CAE, the output from the last convolution-pooling block is usually

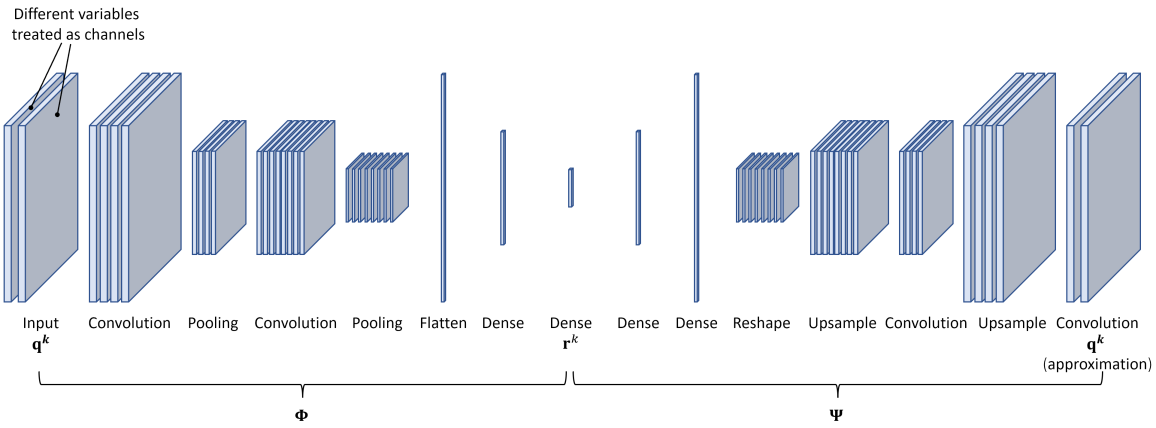


Figure 3.1: Sample CAE architecture. The leftmost and rightmost slabs represent a spatial field.

flattened into a 1D vector. Dense layers are then appended to generate a reduced order variable of arbitrary size.

The decoder $\boldsymbol{\Psi}$ usually takes an inverse structure of the encoder, with the convolution-pooling blocks replaced by upsampling-convolution blocks. The upsampling operation is the inverse of pooling, which expands the spatial dimension of the input, and fills in the blanks with repeated or zero values.

A schematic of a sample CAE architecture with two convolution-pooling blocks and two dense layers is given in Fig. 3.1. In our applications, the different variables in the flow field data are treated as different channels in the input layer. This treatment enables the network to process an arbitrary number of variables of interest without substantial change in structure.

3.3 Autoregressive Models

Without accessing the FOM scheme, non-intrusive models are purely data-driven, i.e. developed by fitting the dynamics from existing solutions. In this section, autoregressive deep-learning models are introduced, which predict the future states of $\mathbf{r}$ based on its previous values. More formally, an autoregression process can be

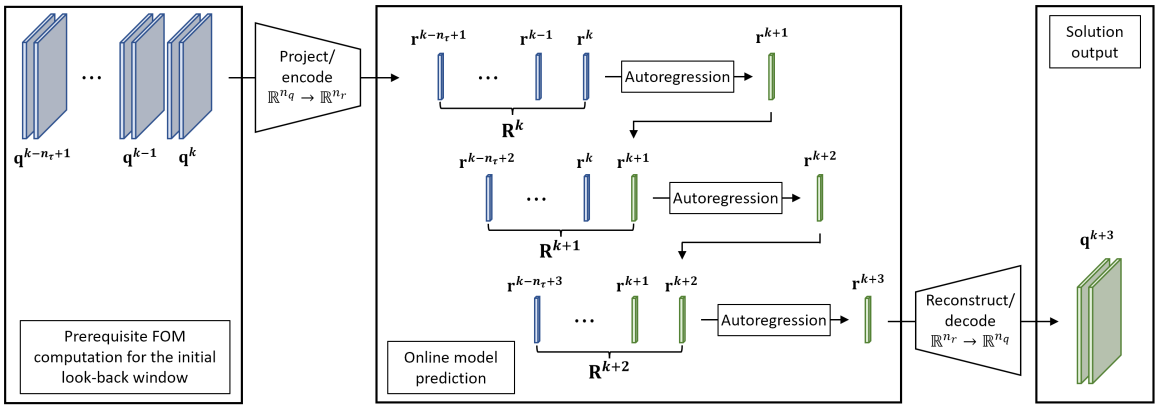


Figure 3.2: Pipeline schematic for a sample autoregressive prediction for 3 future steps beginning at k. Blue slabs: known solution, green slabs: prediction.

expressed using a non-linear function $\mathcal{F} : \mathbb{R}^{n_\tau \times n_r} \to \mathbb{R}^{n_r}$ as:

$$\mathbf{r}^{k+1} = \mathcal{F}(\mathbf{R}_\tau^k), \tag{3.9}$$

where $\mathbf{R}_\tau^k = [\mathbf{r}^{k-n_\tau+1}, \ldots, \mathbf{r}^k]^T$ is called a *look-back window*, which stores n_τ previous values for $\mathbf{r}$ before step k. In an online prediction, $\mathbf{R}_\tau^0$ is initialized with the existing solution, and then gets updated step-by-step, with the latest predicted value appended to the end, and the oldest value popped from the head. A pipeline schematic for the autoregressive prediction is shown in Fig. 3.2. After generating the prerequisite data necessary for the initial look-back window, which is usually included in the training data, the FOM model is no longer accessed. In the online model prediction stage, all computations are performed for the reduced order variables. The POD reconstruction or decoder is only used to output the solution for time-steps of interest.

3.3.1 MLP for a Special Case

In the special case where $n_\tau = 1$, Eq. (3.9) simplifies to

$$\mathbf{r}^{k+1} = \mathcal{F}(\mathbf{r}^k). \tag{3.10}$$

In this case, $\mathcal{F}$ can be modeled by a simple MLP.

3.3.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are designed to process sequences. In contrast to a layer in a feed-forward network, whose output will not be seen by itself, a RNN layer uses its own previous output as an additional input for its next computation. The recurrent process can be formulated as a function $g : \mathbb{R}^{n_h} \to \mathbb{R}^w$:

$$\begin{aligned} g(\mathbf{h}^k; \mathbf{\Theta}) &= \sigma_o(\mathbf{W}_{oa}\mathbf{a}^k + \mathbf{b}_o), \\ \mathbf{a}^k &= \sigma_a(\mathbf{W}_{aa}\mathbf{a}^{k-1} + \mathbf{W}_{ah}\mathbf{h}^k + \mathbf{b}_a), \end{aligned} \tag{3.11}$$

where $1 \leq k \leq n_\tau$ is the step index in the sequence, which should be distinguished from the time-step index. $\mathbf{a}^k \in \mathbb{R}^{n_a}$ is an intermediate state variable called *activation*, which can be regarded as a latent memory of the previous steps. The shapes of the trainable parameters are $\mathbf{W}_{oa} \in \mathbb{R}^{w \times n_a}, \mathbf{b}_o \in \mathbb{R}^w, \mathbf{W}_{aa} \in \mathbb{R}^{n_a \times n_a}, \mathbf{W}_{ah} \in \mathbb{R}^{n_a \times n_h}, \mathbf{b}_a \in \mathbb{R}^{n_a}$.

Function g swaps through the input sequence, with a zero initial activation $\mathbf{a}^0$, and generates an output sequence of the same length as the input one. For a task as ours, i.e. predict for one step based on a sequence, a RNN works in a *many-to-one* manner. In this case, all hidden layers (layers except the input and the output ones) still output a sequence, which is sent as the input for the next layer. Whereas in the final output layer, only the output for the last step is kept.

3.3.3 Long Short-Term Memory Network

Due to the recurrent computation, gradients become multiplicative in the back-propagation process during training. Thus the gradient vanishing (decaying to zero) and exploding (growing to infinity) phenomena are often encountered by RNNs when processing long sequences.

The Long Short-Term Memory (LSTM) unit is designed to replace the normal recurrent unit Eq. (3.11) in order to solve the gradient problems. Additional latent states, and four *gates* $\mathbf{\Gamma}$, that act as scaling vectors for the latent states are introduced. The gates share an identical architecture with independently trained parameters, given by:

$$\mathbf{\Gamma} = \mathbf{W}_{ah}\mathbf{h}^k + \mathbf{W}_{aa}\mathbf{a}^{k-1} + \mathbf{b}_a. \tag{3.12}$$

The shapes of the parameters are consistent with these for Eq. (3.11).

More specifically, the gates include: an update gate $\mathbf{\Gamma}_u$ that scales the current latent state $\tilde{\mathbf{c}}^k$, a relevance gate $\mathbf{\Gamma}_r$ that scales the previous activation $\mathbf{a}^{k-1}$, a forget gate $\mathbf{\Gamma}_f$ that scales the previous latent state $\mathbf{c}^{k-1}$, and a output gate $\mathbf{\Gamma}_o$ that converts the $\tilde{\mathbf{c}}^k$ to the current activation $\mathbf{a}^k$. The computation for the activation $\mathbf{a}^k$ in a LSTM unit is then given by:

$$\begin{aligned} \tilde{\mathbf{c}}^k &= \tanh(\mathbf{W}_c[\mathbf{\Gamma}_r * \mathbf{a}^{k-1}; \mathbf{h}^k] + \mathbf{b}_c), \\ \mathbf{c}^k &= \mathbf{\Gamma}_u * \tilde{\mathbf{c}}^k + \mathbf{\Gamma}_f * \mathbf{c}^{k-1}, \\ \mathbf{a}^k &= \mathbf{\Gamma}_o * \mathbf{c}^k, \end{aligned} \tag{3.13}$$

where $[\cdot;\cdot]$ denotes an concatenation, $\mathbf{W}_c \in \mathbb{R}^{n_a \times (n_a + n_h)}, \mathbf{b}_c \in \mathbb{R}^{n_a}$.

3.3.4 Temporal Convolutional Network

A Temporal Convolutional Network (TCN) is a special 1D CNN that works on sequential inputs, like our look-back window $\mathbf{R}$. The *receptive field* is an important concept in CNNs, which indicates the range of elements in the input that each output element is dependent on. In standard convolutions, the receptive field grows linearly with the number of layers and the kernel size. This becomes a major disadvantage when applied to long sequential data, leading to a need for an extremely deep network or large kernels. As a solution, dilated 1D convolution is used in TCNs, which is a

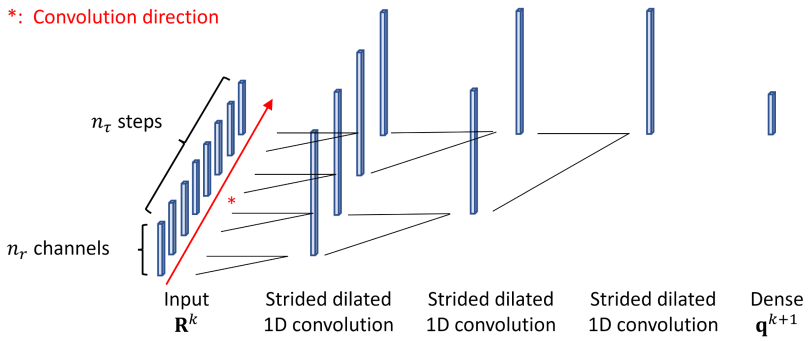


Figure 3.3: Sample TCN architecture, $d = 2, s = 2$.

modified convolution operation with dilated connectivity between the input and the kernel. Around the i-th step in a vector $\mathbf{h}$, the dilated convolution is given by:

$$(\mathbf{h} *_d \mathbf{k})_i = \sum_{p=0}^{w} \mathbf{h}_{i-dp}\mathbf{k}_p, \tag{3.14}$$

where d is called the *dilation order*, and w is the 1D kernel size. By increasing d exponentially with the depth of the network, the receptive field also grows exponentially, thus long sequences can be efficiently processed. A visualization of a TCN with multiple dilated convolutional layers of different dilation orders can be found in *Oord et al.* (2016).

It should be realized that the dilation only affects the kernel connectivity and does not change the output shape, therefore, a sequence processed through it will remain a sequence. To compress the sequence dimension using TCN, the dilated convolution will be performed in a *strided* manner. A stride s refers to the distance between two convolution centers, with $s = 1$ for a standard convolution. The output size of each convolution layer is divided by s, and thus reduces exponentially across the layers. At the last TCN layer, only one number is output for each channel. Additional dense layers can be appended to increase the capacity of the model. A schematic of a sample TCN architecture is shown in Fig. 3.3, where $d = 2, s = 2$ for the dilated convolution layers and one dense layer is used.

3.4 Comparison of Autoregressive ROMs on a Wave Propagation Problem

3.4.1 Problem Statement

To demonstrate the advantage and limitations of the non-intrusive ROMs, a comparison is performed on a 1D supersonic wave propagation problem. A uniform steady state is used as the IC. Unsteady dynamics are then triggered by imposing a perturbation p' on the inlet pressure, given by:

$$p' = 0.01\bar{p}\sin(2\pi 1000t(1 - 0.5e^{-0.005/t})), \tag{3.15}$$

where $\bar{p}$ is the steady-state pressure. The perturbation signal is visualized in Fig. 3.4a. Following the time-dependent function, a pressure wave at a gradually varying frequency is generated and propagates downstream, which can be visualized by the spatial profiles at a few sample time instances in Fig. 3.4b. The task is to predict the future states of such waves based on their history.

The problem is designed to be used for the study of non-intrusive models either in the current "standalone" manner (i.e. without external inputs, purely learning from the history dynamic patterns), or in a model-solver coupling setting as will be demonstrated in Sec. 5.3.4. The idealized design is based on the following considerations:

1. Different scales of dynamics should be present in different periods of the problem, such that the predictive capability of the projection-based dimensionality reduction methods can be evaluated. This is reflected by the different spatial frequencies for the pressure profiles at different time-steps.

2. Due to the mechanism of the autoregressive models, the future states of the dynamics should be learnable from its history. However, at the same time it cannot be purely periodic or linearly non-periodic, otherwise, the development of a

complex model would be meaningless. This is reflected by the time-dependent perturbation function.

3. Impact of a boundary/interface condition should be clearly visible. Within the overall multi-fidelity framework of the thesis, it is important to investigate the characteristics of different methods in response to a boundary/interface condition, especially for the study in Sec. 5.3.4. The present wave propagation problem sets a strong dependency for the internal dynamics on the boundary condition.

3.4.2 Models Details

The problem is governed by the 1D unsteady Euler equation, which is taken as the FOM:

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = 0, \qquad \mathbf{q} = \begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix}, \ \mathbf{F} = \begin{pmatrix} \rho u \\ (\rho u^2 + p) \\ (\rho E + p)\, u \end{pmatrix}. \tag{3.16}$$

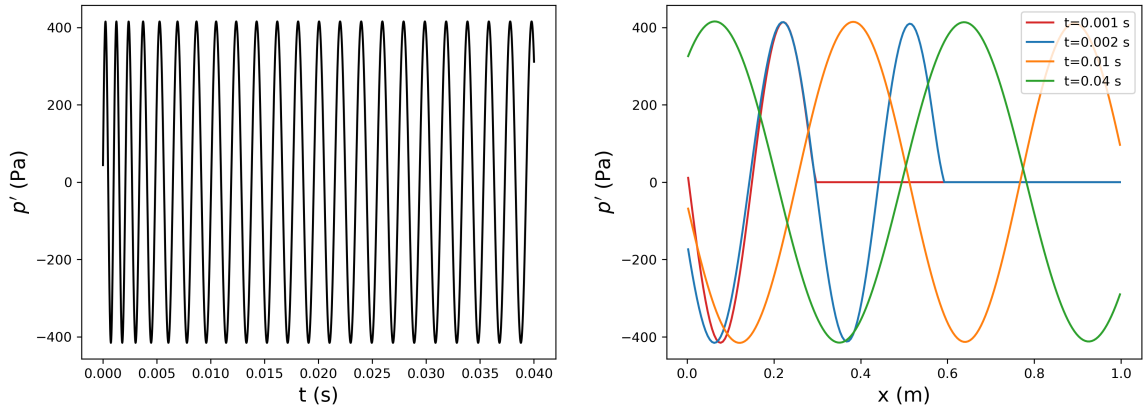
The solution is performed on a 200-cell mesh using a finite volume discretization solver with forward-Euler time-integration at CFL=0.5. For the tests below, data for the first 0.03 s, including 1800 time-steps, is used for training, and the next 0.01 s, including 600 steps, is used for testing.

6 non-intrusive ROMs from the combinations between 2 dimensionality reduction methods (POD, CAE) and 3 autoregressive models (MLP, LSTM, TCN) are studied. The CAE architecture is illustrated in Fig. 3.5, which consists of an encoder with three 1D convolution-pooling blocks followed by one dense layer and a symmetric decoder. The width of the layers scales with n_r. The MLP consists of 4 128-unit

Table 3.1: Autoregressive model details.

Model	n_τ	Width	Depth	Number of parameters	RMSE POD	RMSE CAE
MLP	1	128	4	50688	0.83	0.53
LSTM	25	32	3	21508	0.13	0.11
TCN	25	32	3	67396	0.20	0.22
Galerkin	N/A				0.18	N/A

dense layers, followed by a n_r-unit output dense layer. The two more complex models consist of three 32-unit LSTM/TCN layers, also followed by an output dense layer. More details for the autoregressive models are provided in Table 3.1.



(a) Pressure perturbation signal

(b) Sample spatial profiles

Figure 3.4: Spatial and temporal profiles of the perturbed wave.

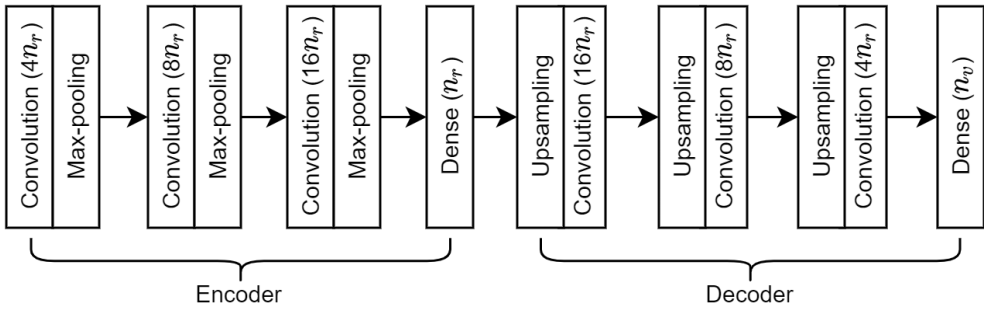


Figure 3.5: CAE architecture.

3.4.3 Offline Projection-Reconstruction

The two dimensionality reduction methods, POD and CAE, are first compared in projection-reconstruction (encoding-decoding) tests at 6 reduced order dimensions $n_r = \{1, 2, 4, 8, 16, 32\}$. The reconstruction fidelity is evaluated using the Root-Mean-Square Error (RMSE). As mentioned in Sec. 1.3.3.2, all variables in the input (also the target output) are scaled to zero mean and unit standard deviation, thus a single joint RMSE can be used, and the results are plotted in Fig. 3.6. It can be observed that at a small n_r, CAE provides a lower error in both the training and the testing stage. Beyond $n_r = 16$, the CAE RMSE saturates and the POD starts to perform better. Actually, at this point the number of trainable parameters in the CAE is on the same order as the number of training samples or even larger, thus the CAE cannot be trained sufficiently.

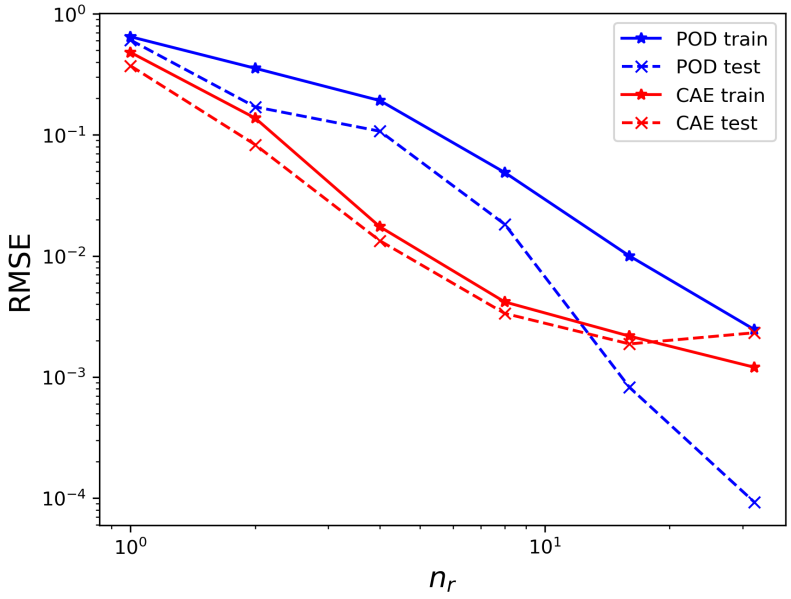


Figure 3.6: Projection-reconstruction RMSE, axes in log scale.

3.4.4 Online Future-State Prediction

In this subsection, online prediction is performed with the 6 combinations of non-intrusive models, in addition to an intrusive POD-Galerkin ROM. A reduced order dimension $n_r = 4$ is used in both the POD and the CAE. When combined with different dimensionality reduction methods, the models are independently trained. The last n_τ steps of the $\mathbf{Q}_r$ from the training period are used as the initial condition for the rollout computation.

Again, RMSE on the scaled variables is used as the accuracy metric. In addition, the predicted unsteady response in density ρ at the center of the domain is also plotted in Fig. 3.7. In the POD-based group of methods, we can see that the LSTM provides the lowest error, which is even lower than that for the intrusive POD-Galerkin. The LSTM also has the smallest number of trainable parameters. The TCN result is close to the POD-Galerkin, and both are significantly better than the MLP. It should be reminded that in this case, no boundary condition is accessible to the non-intrusive models, thus they are not directly informed of the change of frequency for inlet perturbation. With a moderately sized look-back window $n_\tau = 25$ or approximately 20% of a cycle, the LSTM and the TCN are able to learn and predict the change dominated by Eq. (3.15). Switching to the CAE, a decrease in the error is observed for the MLP and LSTM, partly due to a lower projection error. The performances for the POD-TCN and the CAE-TCN are similar.

3.5 Multi-Level Convolutional Autoencoder Networks for Parametric Prediction of Spatio-Temporal Dynamics

In many applications, the task is not to predict for the future states following a known state, but instead, is to predict the spatio-temporal field at different parameters for the same span as existing data. In this case, the design of non-intrusive ROMs

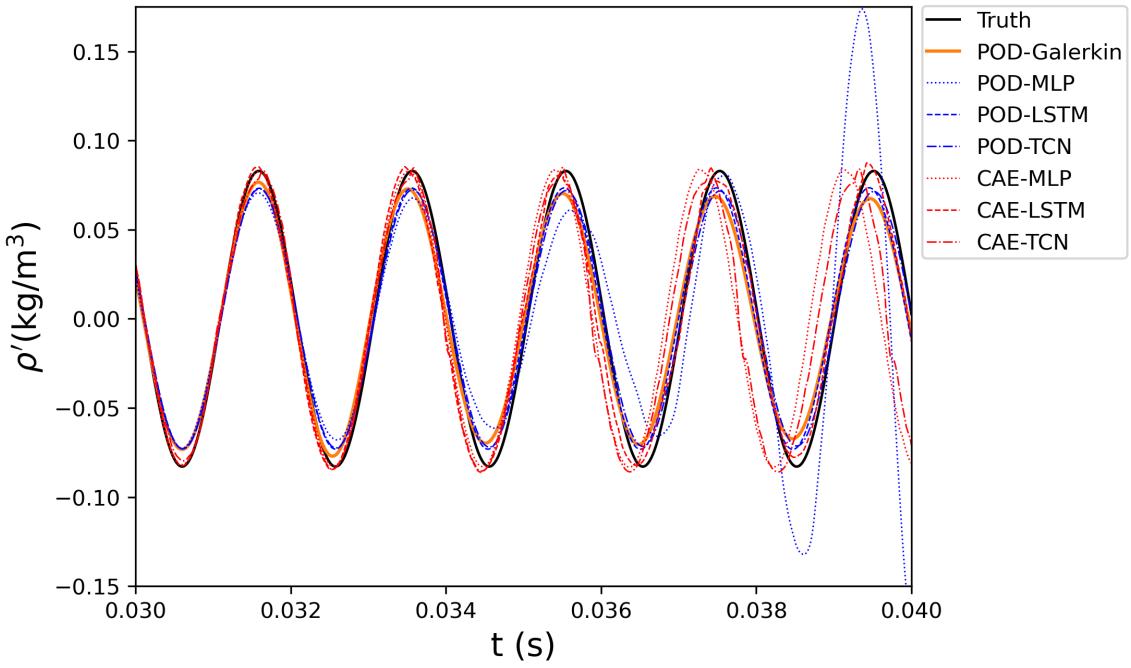


Figure 3.7: Predicted unsteady response in ρ.

can be more flexible and deviate further from the intrusive ones. In this section, we propose a model consisting of multi-level autoencoder networks for the parametric prediction of spatio-temporal dynamics in a "one-shot" manner, i.e. in one inference procedure, solutions for all time-steps of interest are computed.

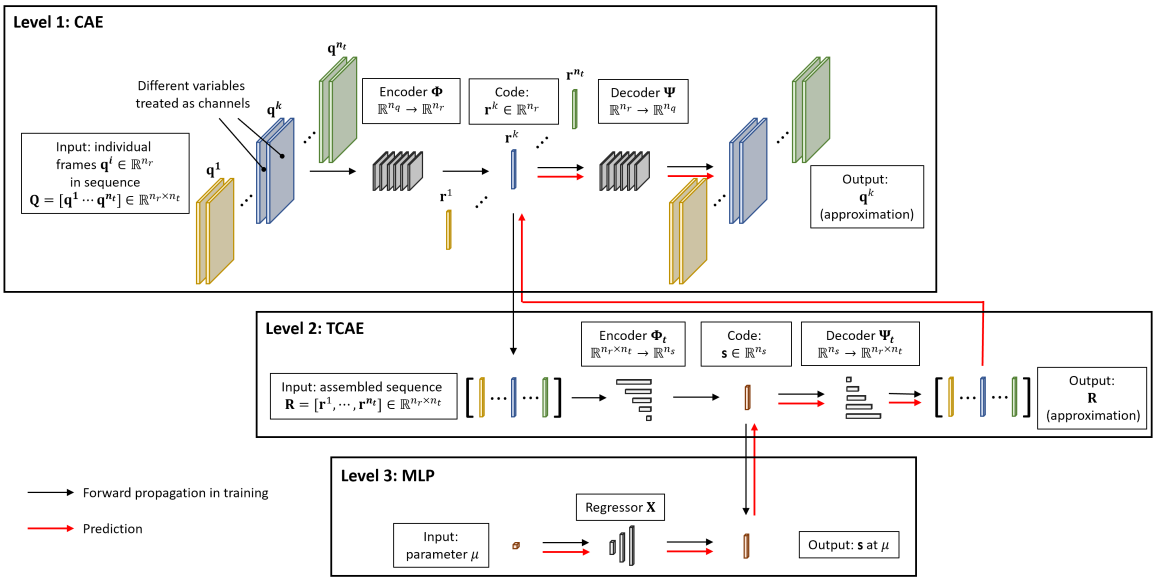


Figure 3.8: Model pipeline for multi-level autoencoder networks.

3.5.1 Constitute Levels

As outlined in Fig. 3.8, the model has the following three levels of networks.

3.5.1.1 CAE

The CAE serves the same purpose as it does as a leading level to the autoregressive models introduced in Sec. 3.3 – to perform spatial compression/decompression of the flow field frame-by-frame. In this setting, the encoding and decoding processes, Eq. (3.3) and Eq. (3.4), can be simplified as:

$$\mathbf{r}^k = \mathbf{\Phi}(\mathbf{q}^k), \tag{3.17}$$

$$\mathbf{q}^k = \mathbf{\Phi}(\mathbf{r}^k). \tag{3.18}$$

3.5.1.2 Temporal Convolutional Autoencoder

For a spatio-temporal collection of snapshots, $\mathbf{Q} \in \mathbb{R}^{n_q \times n_t}$, the frame-by-frame encoded outputs from a CAE encoder can be assembled as $\mathbf{R} = [\mathbf{r}^1, \ldots, \mathbf{r}^{n_t}]^T$. This is similar to the look-back window $\mathbf{R}_\tau$ used in the autoregressive models, but with a complete set of n_t steps, instead of the n_τ look-back steps only. In practical problems, n_t can be $O(10^3)$ or beyond, thus the total DoF in $\mathbf{R}$ is still large and hard to be processed by a single model efficiently. In our model, a Temporal Convolutional Autoencoder (TCAE) is used to further compress/decompress the temporal dimension of $\mathbf{R}$. The encoding and decoding processes are given by:

$$\mathbf{s} = \mathbf{\Phi}_t(\mathbf{R}), \tag{3.19}$$

$$\mathbf{R} = \mathbf{\Psi}_t(\mathbf{s}), \tag{3.20}$$

where we use the sub-script t to distinguish the TCAE encoder/decoder from the CAE ones.

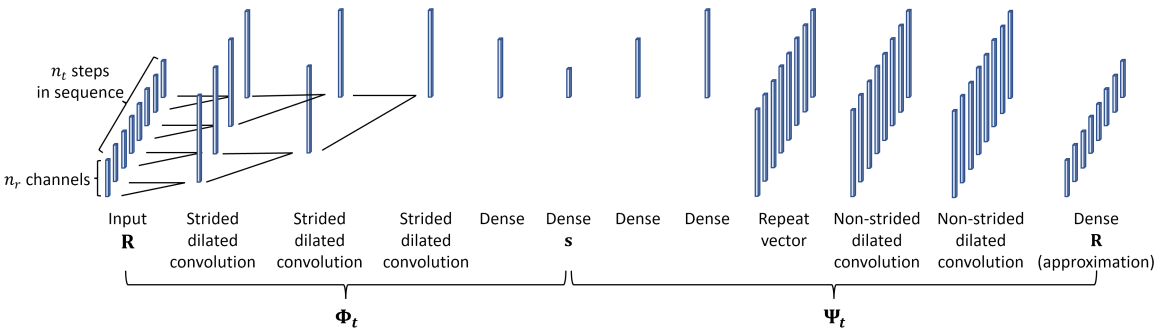


Figure 3.9: Sample TCAE architecture.

A schematic of a sample TCAE architecture is shown in Fig. 3.9. The encoder $\mathbf{\Phi}_t : \mathbb{R}^{n_r \times n_t} \to \mathbb{R}^{n_s}$ is essentially a TCN by itself, where $\mathbf{R}$ is processed as a sequence of length n_t with n_r channels. For each of the channels, strided dilated convolutions are performed along the temporal dimension, and all steps in the sequence are integrated into one number at the last convolution layer. At this point, the temporal dimension is eliminated. After the convolution layers, several dense layers are used to further compress the intermediate variable into the output code vector $\mathbf{s} \in \mathbb{R}^{n_s}$. The decoder $\mathbf{\Psi}_t : \mathbb{R}^{n_s} \to \mathbb{R}^{n_r \times n_t}$ is in general an inverted counter-part of $\mathbf{\Phi}_t$, but with non-strided dilated convolution layers, and with a dense layer added to the end as the output layer.

3.5.1.3 MLP

After two levels of encoding, the output $\mathbf{s}$ is an efficient representation for the full spatio-temporal field. If $\mathbf{s}$ can be predicted at a new condition, the corresponding flow field at that condition can then be obtained by performing two levels of decoding using $\mathbf{\Phi}_t$ and $\mathbf{\Phi}$. In this case, the DoF to be predicted is reduced from $n_q \times n_t$ to n_s. In this model, we assume that the condition of interest can be described by a set of parameter(s) $\mu \in \mathbb{R}^{n_\mu}$, which can be either a scalar or a vector, and that $\mathbf{s}$ is a function of μ. To learn the function, nonlinear regression is performed in the third level $\mathbf{X} : \mathbb{R}^{n_\mu} \to \mathbb{R}^{n_s}$ of the model, which is a MLP by itself.

Finally, for an unseen parameter μ^*, a prediction for the complete n_t steps of the corresponding flow field is given by:

$$\begin{aligned} \mathbf{R}^* &= \mathbf{\Psi}_t(\mathbf{X}(\mu^*)), \\ \mathbf{Q}^* &= [\mathbf{\Psi}(\mathbf{r}^*) \text{ for } \mathbf{r}^* \in \mathbf{R}^*]. \end{aligned} \tag{3.21}$$

In practice, the spatial decoding using $\mathbf{\Psi}$ can be performed on selected time-steps of interest only, which will further reduce the computation cost.

3.5.2 Training Procedure

The constitute networks are trained sequentially. This choice avoids the training of an overly deep network, and also makes the detailed design for each level easier. We consider n_{train} (should be distinguished from n_μ, which is the size of μ for each case) sets of training data provided, each being a set of spatio-temporal snapshots at a different μ. For the CAE, every frame $\mathbf{q}$ in each of the training sets is taken as a training input as well as the corresponding target output, resulting in $n_{\text{train}} \times n_t$ training samples in total.

After the CAE is trained, the encoder Φ is used to encode all training snapshots. The outputs for each μ is arranged into an individual sequence $\mathbf{R}_\mu$, and used as a sample input/target output for the TCAE, resulting in n_{train} training samples in total.

Finally, after the TCAE is trained, Φ_t is used to encode each $\mathbf{R}_\mu$ into a vector $\mathbf{s}_\mu$, which is used as the target output for the MLP $\mathbf{X}$ for the corresponding input μ. The total number of training samples is also n_{train}.

3.6 Parametric Prediction for Flow Over a Cylinder

3.6.1 Problem Statement

In this test, transient flow over a cylinder is considered. Despite the simple two-dimensional flow configuration, the complexity is characterized by the evolution of the flow from non-physical, attached flow initial conditions to boundary layer separation and consequently, vortex shedding. The flow dynamics is determined by a global parameter, the Reynolds number Re, which is taken as the parameter of interest μ, defined as :

$$Re = \frac{U_\infty D}{\nu}, \tag{3.22}$$

where U_∞ is the inflow velocity, D is the diameter of the cylinder and ν is the viscosity of the fluid. Solutions for x-velocity u and y-velocity v at 6 different $Re_{\text{train}} = \{125, 150, 175, 225, 250, 275\}$ are used as the training data, and prediction is performed at $Re_{\text{test}} = 200$ for the same variables.

For every Re, the solution is started from the same IC corresponding to $Re = 20$. Depending on Re, the unsteady vortex shedding initializes and develops at different rates, and finally settles into Limit Cycle Oscillation (LCO) at different frequencies. To cover the full transition process at all Re studied, a large number of frames, $n_t = 2500$, is included in each set of snapshots. In every frame, the solution to the incompressible Navier-Stokes equations is interpolated onto a 384×192 uniform grid, spanning a domain of $20D$ in the x-direction and $10D$ in the y-direction. A sample contour of the flow field at the final step $k = 2500$ at the testing parameter $Re = 200$ is given in Fig. 3.10a. To better illustrate the transition process, as well as the difference between different Re, a point monitor is placed $1D$ downstream of the center of the cylinder. The velocity at this location is shown for two training parameters $Re = \{125, 275\}$ and the testing parameter $Re = 200$ in Fig. 3.11a.

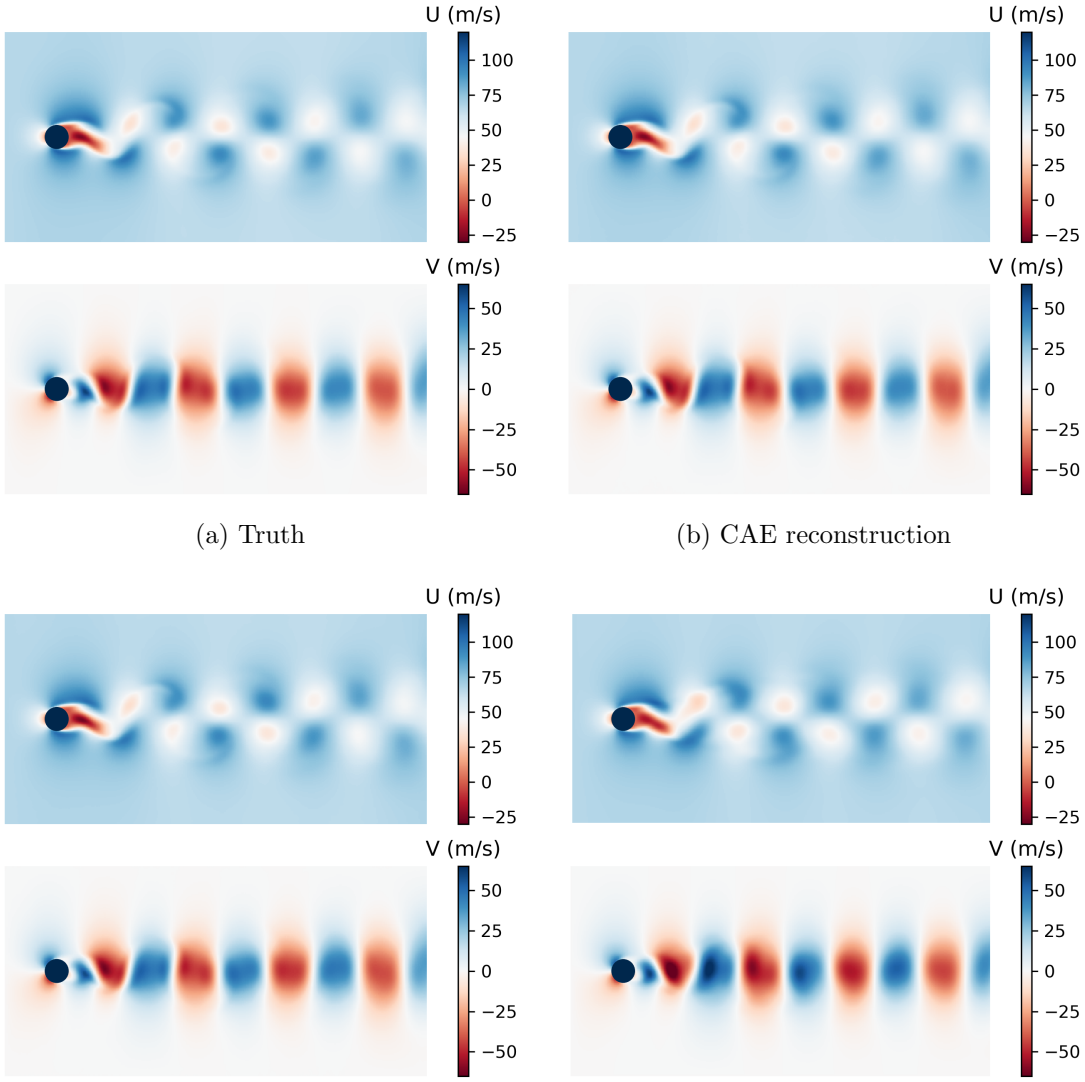
3.6.2 Results

In the model used for the numerical test, the CAE includes 4 convolutional layers and 1 dense layer in the encoder, and a symmetrically arranged set of layers in the decoder. The total input DoF to the CAE is $n_q = 384 \times 192 \times 2 = 147456$, which is encoded into a latent size $n_r = 60$. A qualitative evaluation of the CAE performance in the testing condition is provided by the contours in Fig. 3.10b for the reconstruction result $\mathbf{\Psi}(\mathbf{\Phi}(\mathbf{q}))$, as well as the probed history in Fig. 3.11b. In both figures, deviation from the truth can be hardly observed.

The TCAE includes 12 dilated convolutional layers followed by 2 dense layers in the encoder, and 3 dense layers followed by 12 convolutional layers in the decoder. The input $\mathbf{R} \in \mathbb{R}^{2500\times60}$ is encoded to a latent size $n_s = 100$. For the last level, a MLP of 3 dense layers is used. The reconstruction and prediction results for the intermediate variables are visualized in Fig. 3.12. Except for the oscillating noise in the initial period of $\mathbf{R}$, the results also follow the truths closely. The results are especially encouraging considering that only 6 training samples are available for the TCAE and the MLP. Based on the results for the constitute levels, it is reasonable to expect a well-predicted result from the framework, as also visualized in Fig. 3.10b and 3.11b.

In addition, POD is used to replace the CAE as the top level using the same number of modes $n_r = 60$, and the numerical tests are repeated with independently trained following levels. The prediction result is visualized in Fig. 3.10d, where a noticeable deviation from the truth and the CAE-based results is observed, characterized by blurred vortex structures and over-predicted magnitudes.

Quantitatively, percentage errors in different stages of output, each normalized by the maximum absolute value of the corresponding truth, are summarized in Table 3.2. Error for the predicted velocity components is below 1.5%, and the maximum error in all stages is below 2%. With a larger number of training samples than other levels, the



(a) Truth

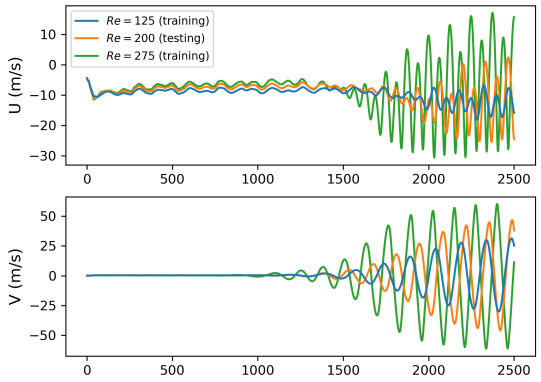
(b) CAE reconstruction

(c) Framework prediction

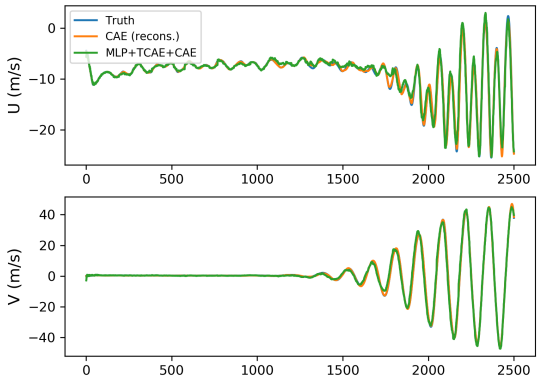
(d) POD-based framework prediction

Figure 3.10: Comparison of flow fields at $k = 2500, Re = 200$.

error for the CAE is especially low, which is below 0.2%. The POD-based results are also provided in the same table, which is around 3 times larger in the reconstruction stages, and 1.5 times larger in the prediction. Finally, timing results for the training and inference of different parts of the framework are listed in Table 3.3. The total prediction time for 2500 frames is less than 5 s. In comparison, the original CFD simulation is conducted with an in-house solver GEMS (General Equation and Mesh Solver) developed by Purdue University (*Huang et al.*, 2019b), which uses an implicit

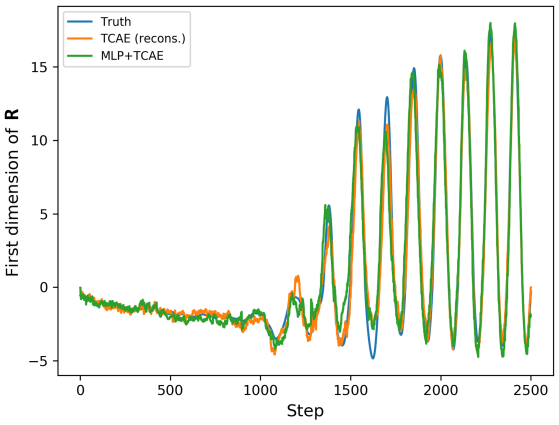


(a) Responses at different Re.

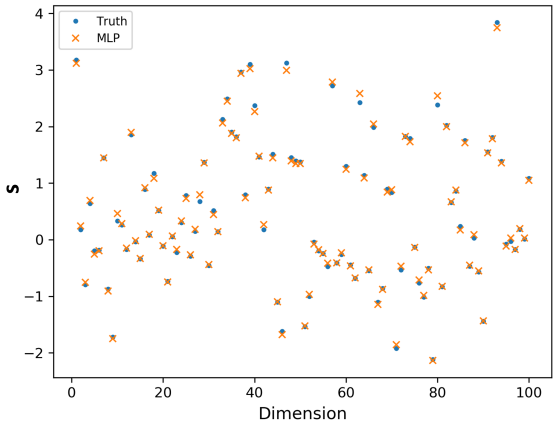


(b) Truth, CAE reconstruction and framework (MLP+TCAE+CAE) prediction for $Re = 200$.

Figure 3.11: Probed variables.



(a) Truth ($\mathbf{\Phi}(\mathbf{Q})$), TCAE reconstruction, and framework (MLP+TCAE) prediction for the first dimension of $\mathbf{R}$



(b) Truth ($\mathbf{\Phi}_t(\mathbf{R})$) and MLP prediction for $\mathbf{s}$.

Figure 3.12: Comparison of intermediate latent variables for $Re = 200$.

time integration method with 10 sub-iterations per time-step. The simulation takes more than 13000 s on an 18-core Intel Xeon Gold 6154 CPU running at 3.70GHz. An acceleration of more than 2500 times is achieved.

3.7 Summary

In this chapter, non-intrusive ROMs based on the combinations of multiple techniques are studied. Led by a few basic neural network layers and architectures, the

Table 3.2: Percentage errors in different stages of output.

Stage	Truth	Output	Percent error
Training reconstruction	$\mathbf{q}\,(u/v)$	$\mathbf{\Psi}(\mathbf{\Phi}(\mathbf{q}))$	0.05%/0.09%
	$\mathbf{R}=\mathbf{\Phi}(\mathbf{Q})$	$\mathbf{\Psi}_t(\mathbf{\Phi}_t(\mathbf{R}))$	0.82%
	$\mathbf{\Phi}_t(\mathbf{R})$	$\mathbf{X}(Re)$	0.56%
POD-based	$\mathbf{q}\,(u/v)$	$\mathbf{V}\mathbf{V}^T(\mathbf{q})$	0.39%/0.41%
Testing reconstruction	$\mathbf{q}\,(u/v)$	$\mathbf{\Psi}(\mathbf{\Phi}(\mathbf{q}))$	0.11%/0.19%
	$\mathbf{R}=\mathbf{\Phi}(\mathbf{Q})$	$\mathbf{\Psi}_t(\mathbf{\Phi}_t(\mathbf{R}))$	1.71%
POD-based	$\mathbf{q}\,(u/v)$	$\mathbf{V}\mathbf{V}^T(\mathbf{q})$	0.38%/0.45%
Prediction	$\mathbf{s}=\mathbf{\Phi}_t(\mathbf{R})$	$\mathbf{s}^*=\mathbf{X}(Re)$	1.02%
	$\mathbf{R}=\mathbf{\Phi}(\mathbf{Q})$	$\mathbf{R}^*=\mathbf{\Psi}_t(\mathbf{s}^*)$	1.99%
	$\mathbf{Q}\,(u/v)$	$\mathbf{\Psi}(\mathbf{R}^*)$	0.68%/1.37%
POD-based	$\mathbf{Q}\,(u/v)$	$\mathbf{V}(\mathbf{R}^*)$	1.01%/1.97%

Table 3.3: Training and inference time.

Training				
Component	CAE	TCAE	MLP	Total
Training epochs	200	1500	2000	-
Computing time (s)	8735	979	84	9798
Prediction				
Component	$\mathbf{\Psi}$ (2500 frames)	$\mathbf{\Psi}_t$	MLP	Total
Computing time (s)	4.55	0.13	0.011	4.691

(convolutional) autoencoder is introduced as an alternative dimensionality reduction method to POD. Through the encoding and decoding processes, an autoencoder tries to compress and reconstruct the network input with the smallest hidden variable. CAE, often interchangeably with POD, serves as the first level for the non-intrusive ROMs introduced in the rest of the chapter. Based on the form of prediction, the subsequent levels are designed differently.

In a closer relation to the intrusive ROM, in which time-integration is necessary, autoregressive models are used to predict the future states of reduced order variables in a rollout (iterative) manner, based on a look-back window. Architectures for several models, mainly the LSTM and the TCN, are introduced. A wave propagation problem is designed to compare the different combinations of dimensionality

reduction methods and the autoregressive models, as well as the baseline intrusive ROM. It is demonstrated that, even without the knowledge of a constantly changing boundary condition, an autoregressive model can learn from the history dynamic patterns, and sometimes perform even better than an intrusive ROM. In the same case, the limitation of models without a long look-back window, i.e. the MLP, is clearly illustrated.

For the direct one-shot parametric prediction of spatio-temporal dynamics, a model based on multi-level autoencoder networks is proposed. The model uses a TCAE to perform an additional level of dimensionality reduction in the temporal dimension. The use of dilated convolutions results in an exponential growth of the reception field, allowing the TCAE to compress long sequences efficiently with a small number of layers. A MLP is used to predict the spatio-temporally compressed code at unseen parameters, from which the prediction for the full spatio-temporal field can be decoded. In the numerical test of predicting transient flow over a cylinder at a new Reynolds number, 2500 frames of the flow field are predicted within 5 seconds, bringing an acceleration of over 2500 times. At the same time, the transition process at a different rate and frequency from any training case is predicted with an error below 1.5%. In this and the previous wave propagation test, the advantage of CAE over POD is also demonstrated with smaller errors in both reconstruction and prediction.

In the larger landscape of this thesis, this chapter initiates the discussion on non-intrusive models. However, only ROMs have been covered so far, which depend on global dimensionality reduction methods and does not solve many problems in the low data limit, such as a lack of portability between different geometries. The CAE even brings an additional requirement of Euclidean data. The coupling method between a non-intrusive model and a solver also remains to be explored. These problems are discussed and addressed in the next two chapters.

CHAPTER IV

Conditional Parameterization and Local Surrogate Models

4.1 Introduction

Despite promising results on canonical problems, commonly used ROMs have inherent limitations. As mentioned in Sec. 1.4, a global dimensionality reduction method, such as POD or autoencoder, generates a fixed mapping between the geometric coordinates and the encoded digits. This limits their portability for new geometries and dynamic patterns. This limitation can be alleviated by local surrogate models such as a purely convolutional CNN (to be distinguished from a standard CAE that uses dense layers to process flattened convolution outputs), however, the latter requires an interpolation of existing data to a structured, Euclidean space, introducing additional cost and error, especially at irregular geometry boundaries. In a broader scope of deep learning models, a few more commonly shared problems also hinders the development of efficient models for physical simulations in the low data regime, such as the frequent concatenation of heterogeneous features like graph node and edge features, or a brute-force fitting of high-order terms. Indeed, even a simple quadratic term needs to be fitted via a number of hidden units with standard ReLU-activated dense layers.

In this chapter, with a focus on mesh-based modeling of physical systems, we demonstrate the advantages of local surrogate models, Conditional Parameterization (CP), and GNNs. With a combination of the three, we propose a Conditionally Parameterized Graph Neural Network (CP-GNet), which effectively models complex physics such as chemical source terms, irregular mesh discretizations, and different types of boundary conditions.

Among the related work mentioned in Sec. 1.4, there are a few pieces that are especially inspiring for the work in this chapter. The Message Passing Neural Network (MPNN) (*Gilmer et al.*, 2017) is a special type of GNN that treats local computations on a graph as messages passed between nodes through edges. Such messages can be designed flexibly into different types of functions, and have largely boosted the variety of GNNs. The Edge Conditioned Convolution (ECC) (*Simonovsky and Komodakis*, 2017) makes the parameters for message-passing functions dependent on edge features, and achieved excellent performance on graphs built from irregular point cloud data. On the application of GNNs for physical simulations, Graph Network-based Simulators (GNS) (*Sanchez-Gonzalez et al.*, 2020) introduced an encoder-processor-decoder architecture for particle-based simulations, which is further extended to mesh-based simulations in the MeshGraphNets (*Pfaff et al.*, 2020).

4.2 Conditional Parameterization (CP)

The idea of CP is to use trainable functions of input parameters to generate the trainable parameters of a neural network. In the inference computation of a standard dense layer Eq. (3.2), the values of the parameters are fixed regardless of the inputs. Thus the performance of the model is largely limited by the interpolation range of training data.

By introducing a parameter vector $\mathbf{p} \in \mathbb{R}^{n_p}$ and a trainable function $\theta(\mathbf{p}) : \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_h \times w}$ that computes the weights $\mathbf{W}$ based on $\mathbf{p}$, the CP modification for Eq. (3.2)

is formulated as:

$$f(\mathbf{h};\theta(\mathbf{p}),\mathbf{b}) = \sigma(\theta(\mathbf{p})\mathbf{h}+\mathbf{b}). \tag{4.1}$$

An easy way to incorporate the formulation into existing neural network frameworks is by making θ a single-layer MLP, the conditionally parameterized dense (CP-Dense) layer can be represented by:

$$f(\mathbf{h},\mathbf{p};\mathbf{\Theta}) = \sigma_h\left(\sigma_\theta\left(\langle\mathbf{W},\mathbf{p}\rangle+\mathbf{B}\right)\mathbf{u}+\mathbf{b}\right), \tag{4.2}$$

where $\langle\cdot,\cdot\rangle$ denotes a tensor product. It should be noted that this would bring a change in the dimensions of weights and biases, which become $\mathbf{W}\in\mathbb{R}^{(n_h\times w)\times n_p}$, $\mathbf{B}\in\mathbb{R}^{n_h\times w}$. When the layer width is kept the same, the total number of trainable parameters increases linearly with the parameter size n_p.

In applications, $\mathbf{p}$ is not limited to an additionally-introduced parameter. When simply taking $\mathbf{h}$ as the parameter for itself, the quadratic terms will be introduced. High-order terms, which are prevalent in physical systems, can be easily modeled using multiple such layers.

4.3 Local CP model: a Proof-of-Concept Demonstration

In this section we demonstrate - as a proof-of-concept problem - 1) the limitations of global projection-based methods in advection-dominated problems, and 2) how discretized PDE terms can be fitted exactly with simple CP layers in the solution of the 2D advection-diffusion equation. With periodic boundary conditions, the governing equations are:

$$\begin{aligned} \frac{\partial u(x,y,t)}{\partial t} + \mathbf{a}\nabla u(x,y,t) - \nu\nabla^2 u(x,y,t) = 0, \\ x \in [0,W], y \in [0,H], t \in [0,t_{\text{end}}], \\ u(0,y,t) = u(W,y,t), u(x,0,t) = u(x,H,t), \end{aligned} \tag{4.3}$$

where u is the quantity of interest, such as the mass or temperature, $\mathbf{a} = [a_x, a_y]^T$ is the advection velocity vector at which u is transferred, and ν is the diffusion coefficient. The initial condition (IC) has a rectangular frame with $u = 1$ in the center, and $u = 0$ elsewhere. The Ground Truth (GT) for the test is generated with the first-order upwind finite difference discretization and Forward Euler (FE) time integration, given by:

$$\begin{aligned} &\frac{\Delta u^k_{i,j}}{\Delta t} + \frac{a_x + |a_x|}{2\Delta x}(u^k_{i,j} - u^k_{i-1,j}) + \frac{a_x - |a_x|}{2\Delta x}(u^k_{i+1,j} - u^k_{i1,j}) + \frac{a_y + |a_y|}{2\Delta y}(u^k_{i,j} - u^k_{i,j-1}) \\ &+ \frac{a_y - |a_y|}{2\Delta y}(u^k_{i,j+1} - u^k_{i,j}) - \nu\frac{u^k_{i-1,j} - 2u^k_{i,j} + u^k_{i+1,j}}{\Delta x^2} - \nu\frac{u^k_{i,j-1} - 2u^k_{i,j} + u^k_{i,j+1}}{\Delta y^2} = 0, \end{aligned} \tag{4.4}$$

where i and j are the grid indices in the x and y directions, respectively, and Δx and Δy are the distances between grid points. A sample solution for $\Delta x = 0.02, \Delta y = 0.02, \mathbf{a} = [1.2, 1.2]^T, \nu = 0.035$ is visualized in Fig. 4.1.

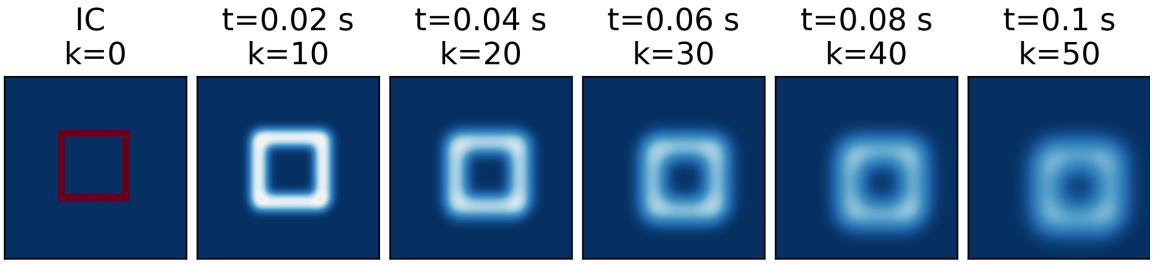
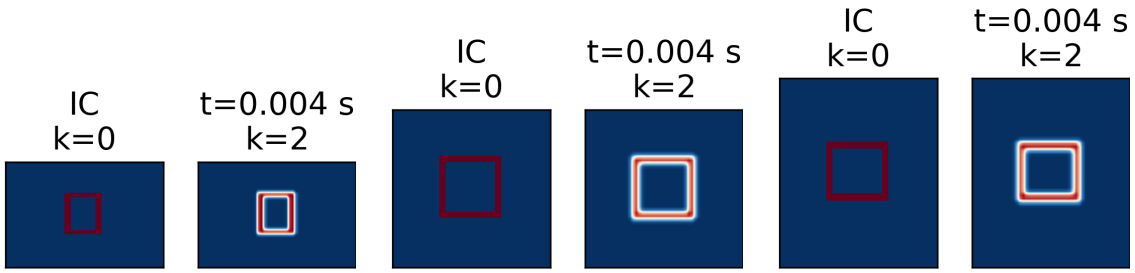


Figure 4.1: Sample solution for $\Delta x = 0.02, \Delta y = 0.02, \mathbf{a} = [1.2, 1.2]^T, \nu = 0.035$.

In common data-driven applications, a large amount of training data is required to train the model sufficiently, and to avoid overfitting. In this test case, however, we

assess the ability of the model to approximate the truth at a machine precision level in the limit of extremely sparse data snapshots. As a demonstration, we use only 3 sets of training data, each for a different set of parameters $\{\Delta x, \Delta y, \mathbf{a}, \nu\}$ to train a model represented by Eq. (4.5). All of the training cases are present in Fig. 4.2 – each set only has 3 snapshots: the IC and the solutions for the initial 2 steps. Testing is performed at a new initial condition and set of parameters outside the interpolation range of training ones, $\Delta x = 0.02, \Delta y = 0.016, \mathbf{a} = [-1.5, 1.5]^T, \nu = 0.02$. The testing case and results are present in Fig. 4.3.



(a) $\Delta x = 0.03, \Delta y = 0.02$, $\mathbf{a} = [1, -0.8]^T, \nu = 0.035$

(b) $\Delta x = 0.02, \Delta y = 0.02$, $\mathbf{a} = [1.2, 1.2]^T, \nu = 0.035$

(c) $\Delta x = 0.02, \Delta y = 0.024$, $\mathbf{a} = [-1, 1]^T, \nu = 0.04$

Figure 4.2: Training data. Each case consists of only the IC and two solution steps.

4.3.1 Limitations of Global Projection-Reconstruction

We first demonstrate the limitations of the global projection-based methods in such an advection-dominated problem. Regardless of the intrusive or non-intrusive model used, the predictive capability of such methods is within an upper limit determined by the reduced-order subspace, which can be quantified by the projection-reconstruction result. In this case of a small number of training snapshots, the number of the total POD modes available is limited to the same number, $n_r = 9$, all of which are used in the projection-reconstruction. In Fig. 4.3 it is shown that the POD is unable to capture either the advection or the diffusion (except for a decay in the maximum magnitude). This can be clearly explained by the POD modes visualized

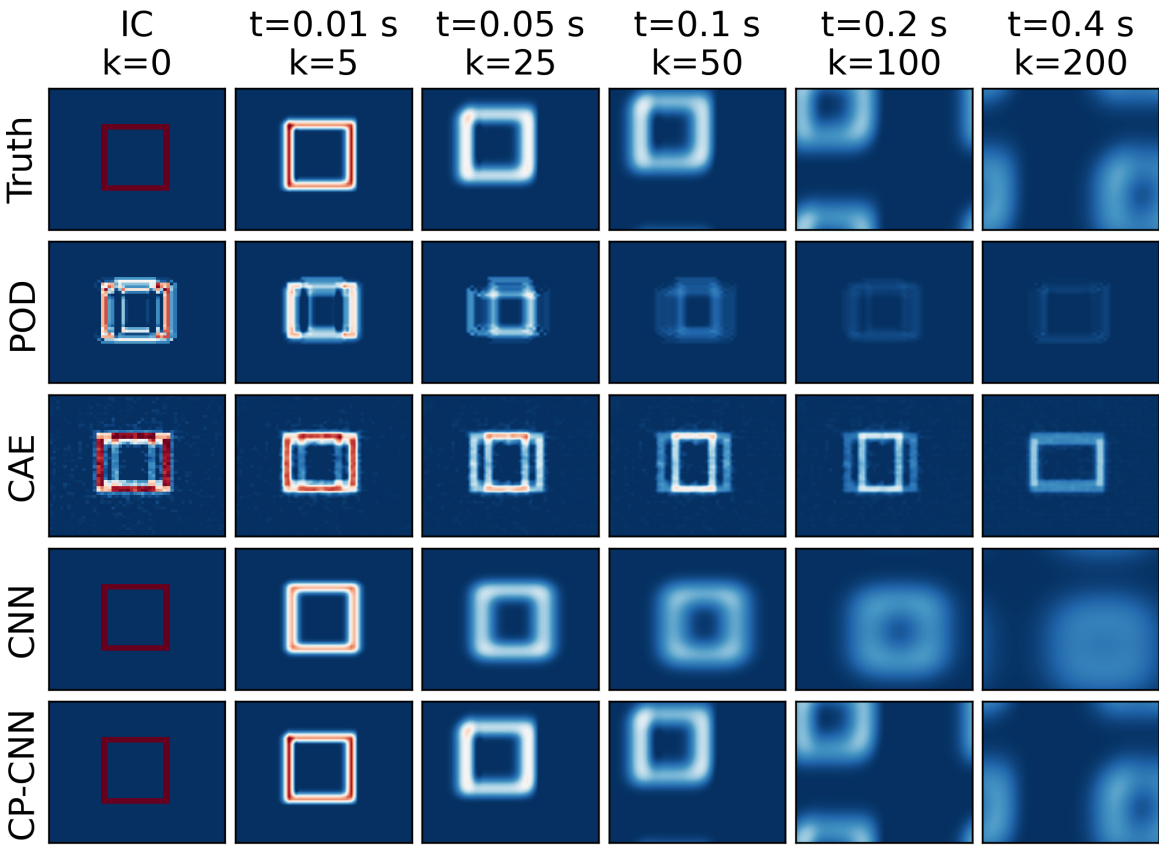


Figure 4.3: Projection-reconstruction (POD, CAE) and prediction (CNN, CP-CNN) results for unseen parameters $\Delta x = 0.02, \Delta y = 0.016, \mathbf{a} = [-1.5, 1.5]^T, \nu = 0.02$. CP-CNN fits the discretized model exactly.

in Fig. 4.4. The POD, and all global model order reduction methods, learn a fixed mapping between the dynamics and the global coordinates (grid indices i, j in this case). To be expressive for dynamics at a specific location of the domain, a representative feature must be present in the training data at the same location. However in this test, we only included the two solution steps in the training, in which the informative features are limited to a small scope around the initial location, and as a consequence, the POD modes only contain information within this scope. Due to the different grid sizes, even the IC is not present at the same grid indices across different cases and therefore cannot be correctly reconstructed.

A 4-layer CAE, which is a 2D version of the one used in Sec. 3.4 is also tested in a similar manner. Despite the visually different results, the CAE is also not able to

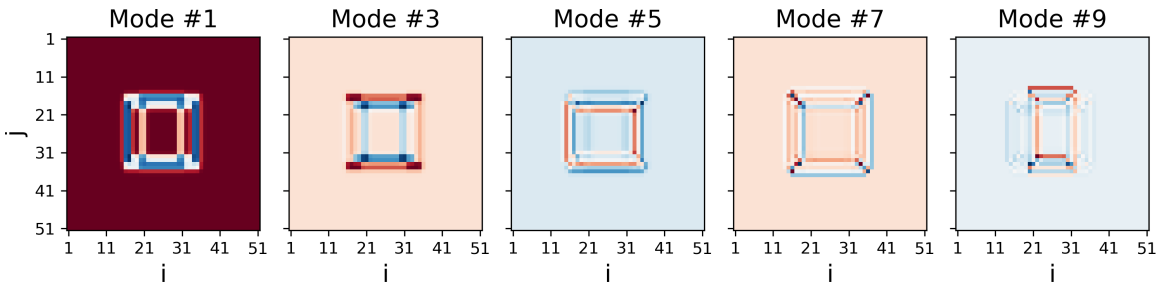


Figure 4.4: Sample POD modes.

reconstruct the testing case correctly. Indeed, CAE faces an additional limitation: to maintain the output size the same as the input through poolings and upsamplings, the input grid size has to be padded or cropped to a exact power of the pooling kernel size. For the results presented, a pooling kernel of 2 is used, and the input is cropped to the center 48×48 grids.

4.3.2 Results for a Local CNN

The vanilla CNN (not in the form of a CAE) is probably the simplest local model: each convolution kernel swipes through the domain and outputs a local output at each stop through the swiping process. The convolution computation itself is not aware of the global coordinates, thus no mapping is built into the model. As an additional remark, the CAE is made a global model due to the use of reshaping and dense layers. A fully-convolutional autoencoder can also be regarded as a local model, but such a model is rarely used in practice due to the lack of flexibility, and a low dimensionality-reduction efficiency.

A CNN model with two convolution layers is trained on the same sparse training data. For a predictive purpose instead of reconstruction, the effective training snapshots is reduced to 2 pairs per case, with the IC and $\mathbf{u}^1$ as the inputs, and $\mathbf{u}^1$ and $\mathbf{u}^2$ as the target outputs. Online rollout prediction is performed for 200 steps from the testing IC. As shown in Fig. 4.3, the CNN is able to predict a quantitatively advection-diffusion process. However, wrong advection velocity and diffusion speed

are predicted, because the model is not parametric to the operating parameters. To achieve an exact fitting, a CP-CNN has to be used.

4.3.3 Exact Fitting with a CP-CNN

Inspired by Eq. (4.4), a CP-CNN model is constructed using a dense layer and two 2D CP-Convolution (CP-Conv) layers in the form:

$$\begin{aligned} \mathbf{h} &= \mathrm{ReLU}(\mathbf{W}_1\mathbf{a}), \\ \Delta\mathbf{u}^k &= \left\langle \mathbf{W}_2, \frac{\Delta t}{\Delta x}\mathbf{h} \right\rangle * \mathbf{u}^k + \left(\left\langle \mathbf{W}_2, \frac{\Delta t}{\Delta y}\mathbf{h} \right\rangle * (\mathbf{u}^k)^T \right)^T \\ &\quad + \left(\mathbf{W}_3 \frac{\nu \Delta t}{\Delta x^2} \right) * \mathbf{u}^k + \left(\left(\mathbf{W}_3 \frac{\nu \Delta t}{\Delta y^2} \right) * (\mathbf{u}^k)^T \right)^T, \end{aligned} \tag{4.5}$$

where $\mathbf{W}_1 \in \mathbb{R}^{2\times 2}$ is the weight for the dense layer, and $\mathbf{h}$ is the hidden output. $\mathbf{W}_2 \in \mathbb{R}^{3\times 1\times 1\times 2}$ is weight for the first CP-Conv kernel, taking $\frac{\Delta t}{\Delta x}\mathbf{h}$ or $\frac{\Delta t}{\Delta y}\mathbf{h}$ as the condition parameter. $\mathbf{W}_3 \in \mathbb{R}^{3\times 1\times 1\times 1}$ is weight for the second CP-Conv kernel, taking $\frac{\nu\Delta t}{\Delta x^2}$ or $\frac{\nu\Delta t}{\Delta y^2}$ as the condition parameter. $(\cdot * \cdot)$ denotes the convolution operation.

The model is trained with Adam optimizer with the training hyperparameters listed in Table 4.1, and the weights learnt [1] are:

$$\mathbf{W}_1 = \begin{bmatrix} 0.33237486 & 0 \\ 0 & -0.5253752 \end{bmatrix}, \mathbf{W}_2 = \begin{bmatrix} 3.00869074 & 2.38949641 \times 10^{-5} \\ -3.00892553 & -1.90361486 \\ -8.69012577 \times 10^{-5} & 1.90334544 \end{bmatrix}, \mathbf{W}_3 = \begin{bmatrix} 0.99999235 \\ -1.99994342 \\ 1.00001345 \end{bmatrix}.$$

Table 4.1: Training hyper-parameters.

Number of training input-output pairs	6
Grid points per snapshot	51×51
Batch size	1
Initial learning rate	0.1
Final learning rate	0.0003
Number of epochs	10000

The rollout prediction result by the CP-CNN is visually identical to the ground

[1] $\mathbf{W}_2$ and $\mathbf{W}_3$ squeezed for simplicity

truth, with a RMSE less than 1×10^{-4}. Indeed when the weights are substituted into Eq. (4.5), we recover Eq. (4.4) to 4-decimal-point precision, as the ideal weight combination is:

$$\mathbf{W}_1 = \begin{bmatrix} 0.5c_1 & 0 \\ 0 & -0.5c_2 \end{bmatrix}, \mathbf{W}_2 = \begin{bmatrix} 1/c_1 & 0 \\ -1/c_1 & -1/c_2 \\ 0 & 1/c_2 \end{bmatrix}, \mathbf{W}_3 = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix},$$

where $c_1, c_2 \in \mathbb{R}_{\neq 0}$.

It is thus clear that for this class of problems, the utility of conventional deep learning is questionable. The author notes that, in practical and more complex applications, an exact fitting will not be achievable even with CP. However, as an idealized demonstration, this test serves the purpose to show how CP networks can represent functional relations between parameters and variables to reduce the training effort for certain terms, improve accuracy, and offer generalizable predictions.

Although the potential of a local model is demonstrated, not all systems can be easily processed by a CNN. The most fundamental limitation is that a CNN requires image-like uniform Euclidean data, but the geometries for real-life objects are often irregular. In the following sections, we discuss how to build a more generic local model using graphs.

4.4 Graph Representation of a Discretized System

4.4.1 Mapping Between a Mesh and a Graph

As mentioned in Sec. 1.3.1, a computation domain is divided into small discrete elements. The discretized domain is represented by a mesh, and the discrete elements, in the form of closed areas (in 2D) or volumes (in 3D) are called cells (or just "elements", in the FEM community). The shape of a cell is determined by the convex

hull of a set of composing spatial points called vertices[2]. The interface shared by two adjacent cells is called an edge (in 2D) or a face (in 3D). Fig. 4.5 shows an example 2D mesh with these components.

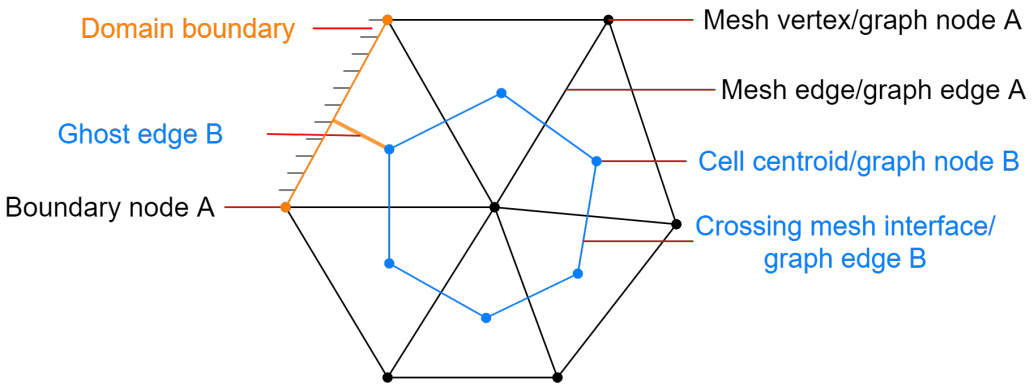


Figure 4.5: Example 2D mesh-to-graph mappings. Graph node/edge A (in black) for the vertex-node mapping, B (in blue) for the cell-node mapping.

With different schemes, the numerical solution can either be stored at the mesh vertices as nodal values (e.g. FDM, node-centered FVM, FEM without interpolation), or at the cell centroids as cell-averaged values (cell-centered FVM). In either case, the discretized system can be mapped to a graph $G(U, E)$, defined by nodes U of size $|U| = n_x$ connected by edges $E \subset U \times U$ of size $|E| = n_e$. For the former, the mesh vertex and edges are directly mapped to the graph nodes and edges. For the latter, each node u_i is located at the corresponding cell centroid $\mathbf{x}_i$, and an edge e_{ij} is defined by the connection between any pair of nodes u_i and u_j that correspond to a pair of neighboring cells.

4.4.2 Node and Edge Features

So far, the graph definition $G(U, E)$ only represents the identification and adjacency information for the discretization. To represent variables, positional information and other information such as boundary conditions, node and edge features need

[2]The mesh vertices are more often called nodes. In this section "vertex" is used to distinguish from a graph node.

to be defined. We use a vector $\mathbf{u}_i$ to denote the node features for node u_i, and a vector $\mathbf{e}_{ij}$ to denote edge features for edge e_{ij}. A graph-based model such as a GNN can then be represented as $\mathcal{G}(G, \mathbf{u}, \mathbf{e})$. We use the physical quantities $\mathbf{q}$ as the node inputs, $\mathbf{u}_i^{\text{in}} = \mathbf{q}_i$. For each edge, distance between the two connected nodes $|\mathbf{x}_i - \mathbf{x}_j|$, and an directional vector $\mathbf{n}_{ij} = (\mathbf{x}_i - \mathbf{x}_j)/|\mathbf{x}_i - \mathbf{x}_j|$ are used as the inputs $\mathbf{e}_{ij}^{\text{in}} = [\mathbf{n}_{ij}; |\mathbf{x}_i - \mathbf{x}_j|]$.

Across different layers of a GNN, $\mathbf{u}$ and $\mathbf{e}$ are updated as hidden variables, and $\mathbf{u}$ for the last layer is taken as the model output. A surrogate model for Eq. (1.2) is then defined by:

$$\Delta \mathbf{q}^k = \mathcal{G}(G, \mathbf{q}^k, \mathbf{e}^{\text{in}}). \tag{4.6}$$

where $\Delta \mathbf{q}^k = \mathbf{q}^{k+1} - \mathbf{q}^k$ is the increment of state variables between two adjacent time steps k and $k+1$.

4.4.3 Representations for Boundary Conditions

The computational domain of a practical problem includes multiple types of boundaries, e.g. the one shown in Fig. 4.5 and the cases in Sec. 4.6.1 & 4.6.2. In classic PDE solvers, they are treated with different boundary conditions, which define explicit formulations to compute relationships of the domain with the external world. For boundaries with known values, such as the Dirichlet, their values can be directly input to the corresponding nodes at every time step. For other types of boundaries that impose certain constraints, such as Neumann and the symmetry boundaries, their conditions and formulations are only defined for the physical quantities, thus cannot be easily transferred for latent variables, especially when multiple layers are used. Therefore, special representations are necessary.

Boundary Node Labels. For the vertex-node mapping, different boundary nodes, corresponding to the boundary vertices in Fig. 4.5, along with the inner nodes, are labeled with *one-hot* encodings, which is a vector, whose size equals the total number

of types of boundaries plus one (for the inner node type). The vector for each node has a single "1" on the digit corresponding to the current node type, and "0" at all other digits. This label is concatenated with $\mathbf{q}$ as the input node features.

Ghost Edges. For the cell-node mapping, the boundary vertices are no longer represented in the graph. Instead, *ghost edges* are introduced. For a cell i with a face lying on a boundary, we introduce a ghost edge vector that connects the corresponding node n_i to the center of the boundary face, as shown in Fig. 4.5. These edges are processed slightly differently from the inner edges, as introduced in Sec. 4.5.2.

4.5 Conditionally Parameterized Graph Neural Networks

4.5.1 The FVM

4.5.1.1 Conservative Equations

The proposed Conditionally Parameterized Graph Neural Network (CP-GNet) model is inspired by the FVM. For a system of conservation laws, Eq. (1.1) can be written in a general form as:

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \vec{\mathbf{F}}(\mathbf{q}) = \mathbf{S}(\mathbf{q}), \tag{4.7}$$

where $\vec{\mathbf{F}}$ is a flux that describes the direction and rate at which the conserved variables pass through the space, and $\mathbf{S}$ is a source term that describes the production or destruction of quantities such as the chemical species in a reacting flow.

The FVM divides the computational domain into contiguous small cells and solve for the cell-averaged values. By applying the Green's law, and assuming that the flux between two cells either is a constant or varies linearly on the interface, the flux term

for a 2D cell i is approximated by:

$$\nabla \cdot \vec{\mathbf{F}}(\mathbf{q})\Big|_i \approx \frac{1}{A_i} \sum_{j \in N(i)} \hat{\mathbf{F}}\left(\mathbf{q}_i, \mathbf{q}_j, \mathbf{n}_{ij}\right) L_{ij}, \tag{4.8}$$

where A_i is the area (volume in 3D) of the cell , $N(i)$ is the neighborhood set of cells around i, L_{ij} is the length (area in 3D) of the shared interface between a pair of neighboring cells. $\hat{\mathbf{F}}\left(\mathbf{q}_i, \mathbf{q}_j, \mathbf{n}_{ij}\right)$ is a complex function involving both cell values as well as the directional vector. A popular choice if the Roe flux *Roe* (1981).

Finally the FVM form of Eq. (4.7) is given by:

$$\frac{d\mathbf{q}_i}{dt} + \frac{1}{A_i} \sum_{j \in N(i)} \hat{\mathbf{F}}\left(\mathbf{q}_i, \mathbf{q}_j, \mathbf{n}_{ij}\right) L_{ij} = \mathbf{S}(\mathbf{q}_i). \tag{4.9}$$

4.5.1.2 Solving for Primitive Variables with Dual Time-Stepping

In many applications, the quantities of interest are *primitive* variables, such as the pressure and the temperature, which are non-conservative. The Dual Time-Stepping (DTS) method enables the FVM to solve for such variables. The DTS introduces an additional pseudo-time-derivative to implicit time-marching schemes. For simplicity, we use the more general form of ODE Eq. (1.2) in place of Eq. (4.9), for which the Backward-Euler scheme with DTS can be written as:

$$\frac{\partial \mathbf{p}}{\partial \tau} + \frac{\mathbf{p} - \mathbf{q}^k}{\Delta t} + \mathbf{f}(\mathbf{p}) = 0, \tag{4.10}$$

where τ is the so-called "pseudo time" or "dual time", and $\mathbf{p}$ is the solution at τ. Eq. (4.10) converts the solution within each physical time step for an unsteady problem into a converging process for a steady problem. The process involves iterative pseudo time-integration of $\mathbf{p}$ with schemes such as the forward-Euler. Each pseudo time step is called a *subitration*, indexed by p, for which the initial stage $\mathbf{p}^0 = \mathbf{q}^k$, and the final stage $\mathbf{p}^P$ converging to $\mathbf{q}^{k+1}$. The choice of P is usually based on preliminary

convergence tests.

To solve for the primitive variables using the conservative equations, Eq. (4.10) is modified as:

$$\Gamma \frac{\partial \mathbf{p}_{\text{prim}}}{\partial \tau} + \frac{\mathbf{p}_{\text{consv}} - \mathbf{q}^k_{\text{consv}}}{\Delta t} + \mathbf{f}(\mathbf{p}_{\text{consv}}) = 0, \tag{4.11}$$

where subscripts "consv" and "prim" are used to distinguish the conservative and the primitive variables, and $\Gamma = \frac{\partial \mathbf{q}_{\text{consv}}}{\partial \mathbf{q}_{\text{prim}}}$ is a Jacobian matrix that transforms the derivatives of primitive variables into the conservative ones, such that the rest of the conservative equation can be maintained.

4.5.2 Architecture

The architecture for the CP-GNet takes an encoder-processor-decoder form, similar to the ones introduced by *Sanchez-Gonzalez et al.* (2020) and *Pfaff et al.* (2020). To be distinguished from the encoder and decoder in an autoencoder that seek global dimensionality reduction and reconstruction, the encoder and decoder in this architecture perform local conversions between physical variables and the latent ones that are used in message passing. A schematic is provided in Fig. 4.6, in which a cell-node mesh-to-graph mapping is used as an example. In practice, the type of mapping can be selected based on the form of data provided (see 4.4.1). The input node feature $\mathbf{q}$ can also be flexibly selected between the conservative and the primitive ones, without further modifications in the network architecture.

4.5.2.1 Encoder

The flux and the source terms in Eq. (4.9) require the processing of arbitrarily complex interactions between mesh elements. While large MLP architectures can represent this complexity, the data requirements to reliably train such networks might be large. In contrast, our proposed encoder takes two CP-Dense layers, taking the output from the previous layer as both the input and the conditional parameter.

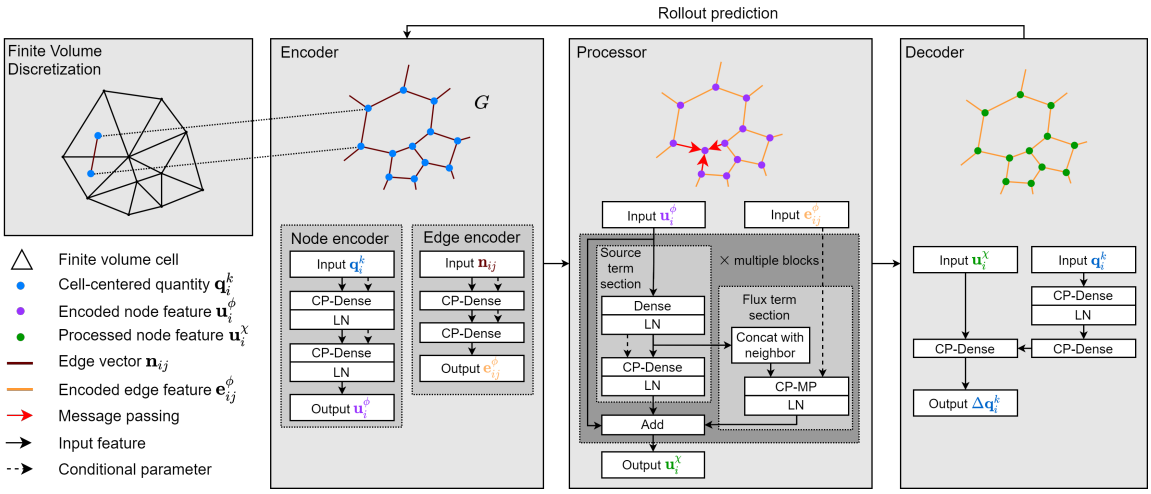


Figure 4.6: Schematic of CP-GNet architecture

Through the encoder, high-order interactions can be easily extracted, allowing a degree of extrapolation by virtue of linearity. The CP-GNet uses two separate, but similarly constructed encoders to process the input node and edge features independently. The output encoded node and edge features are denoted as $\mathbf{u}^\phi$ and $\mathbf{e}^\phi$, respectively. When ghost edges are present, multiple identical sets of edge encoders with independent weights are created, each processing one type of ghost edge, or the inner edges individually.

4.5.2.2 Processor

The processor is a stack of multiple identical blocks with independent weights, with residual connections (*He et al.*, 2016) added between the blocks. As shown in Fig. 4.6, each block includes a "source term" section and a "flux term" section.

The source term section consists of a dense layer followed by a CP-Dense layer, which processes the output from a previous encoder or processor block, and keeps extracting high-order terms through the blocks.

The flux function $\hat{\mathbf{F}}$ in Eq. (4.9) can be approximated by a message-passing process between adjacent nodes. A standard message-passing layer can be regarded as a stack of two dense layers. The first dense layer takes the concatenation of the node features

on both ends of an edge, along with the edge feature itself, as the input, and outputs an edge hidden state. For each node, the hidden states for all the connected edges are summed input a node hidden state $\mathbf{s}$, which is then concatenated with $\mathbf{u}$ on the same node as the input for the second dense layer. The output nodal variable $\mathbf{h}^{\mathrm{MP}}$ from the second layer is used as the result for the message passing. The process can be expressed as:

$$\begin{aligned} \mathbf{s}_i &= \sum_{j \in N(i)} \sigma\left(\langle \mathbf{W}_1, [\mathbf{u}_i; \mathbf{u}_j; \mathbf{e}_{ij}] \rangle + \mathbf{b}_1\right), \\ \mathbf{h}_i^{\mathrm{MP}} &= \sigma\left(\langle \mathbf{W}_2, [\mathbf{u}_i; \mathbf{s}_i] \rangle + \mathbf{b}_2\right). \end{aligned} \tag{4.12}$$

In Eq. (4.12), the hierarchical relation between edge and node features are ignored, and they are concatenated in a "homogeneous" manner. A large number of parameters are required in the dense layers to process the concatenation. In contrast, CP Message-Passing (CP-MP) reserves the relation, and takes the edge feature as the condition parameter for the weights that acts on the node features. Modified from the Edge Conditioned Convolution (ECC) (*Simonovsky and Komodakis*, 2017), the CP-MP computation is formulated as:

$$\begin{aligned} \mathbf{W}_{ij} &= \sigma\left(\langle \mathbf{W}, \mathbf{e}_{ij} \rangle + \mathbf{B}\right), \\ \mathbf{h}_i^{\mathrm{CPMP}} &= \sum_{j \in N(i)} w_{ij} \sigma\left(\langle \mathbf{W}_{ij}, [\mathbf{u}_i; \mathbf{u}_j] \rangle\right), \end{aligned} \tag{4.13}$$

where $w_{ij} = L_{ij}/A_i$ is the flux weight from Eq. (4.9).

For the ghost edges in the cell-node mapping, the CP-MP layer is slightly modified. More specifically, the concatenation $[\mathbf{u}_i; \mathbf{u}_j]$ in Eq. (4.13) is replaced by $\mathbf{u}_i$ only. And for each type of boundary, the weights for the CP-MP layer are trained independently, to let the model learn different types of boundary conditions for the latent variables.

The outputs from the two sections are added as the processor block output. It can be realized that only the node features are updated through the blocks, thus the

edge feature in Eq. (4.13) remains $\mathbf{e}^{\phi}$ across all blocks. The final output node feature from the last block is denoted as $\mathbf{u}^{\chi}$.

4.5.2.3 Decoder

It is common in a PDE solver to use a Jacobian matrix $\Gamma = \partial\mathbf{u}/\partial\mathbf{h}$ to transform the variable increments as $\Delta\mathbf{u} = \Gamma\Delta\mathbf{h}$, such as in Eq. (4.11). The decoder in the GP-GNet serves a similar purpose – to convert hidden variables to the output on the physical space. Similar to the encoder, the decoder consists of three conditionally parameterized dense layers. The first two layers can actually be viewed as a dedicated "encoder" that is similar to the initial node encoder, taking $\mathbf{q}^k$ as the input, but with independent weights. The purpose of this "encoder" is to extract a final conditional parameter, which is used in the third CP-Dense layer in the decoder to determine the weights for the output $\mathbf{u}^{\chi}$ from the processor. The third decoder layer is also the final layer of the model, which outputs the prediction for $\Delta\mathbf{q}^k$.

Except for the edge encoder and the last two layers in the decoder, all dense, CP-Dense, CP-MP layers are appended with LayerNormalization (LN) layers to reduce overfitting and accelerate training.

4.6 Applications of CP-GNet

4.6.1 Reacting Flow in A Rocket Engine Injector

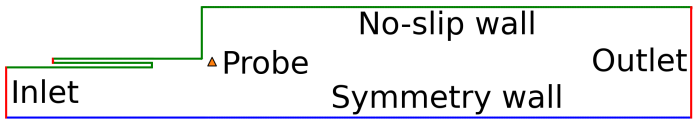
We use a highly complex public dataset (*Huang et al.*, 2019b) as a model of combustion processes in a rocket engine injector (*Huang et al.*, 2021). The ground truth simulation is performed using the finite-volume based General Equation and Mesh Solver (GEMS) (*Harvazinski et al.*, 2015) on a 2D 38523-cell mesh. 8 primitive variables are solved at each discretized cell: $\mathbf{q} = [p, u, v, T, Y_{\mathrm{CH4}}, Y_{\mathrm{O2}}, Y_{\mathrm{H2O}}, Y_{\mathrm{CO4}}]^T$, where p is the pressure, u and v are the x- and y-velocity components, T is the temperature

and $\{Y_{\mathrm{CH4}}, Y_{\mathrm{O2}}, Y_{\mathrm{H2O}}, Y_{\mathrm{CO4}}\}$ are the mass fractions for the chemical species involved in the combustion process.

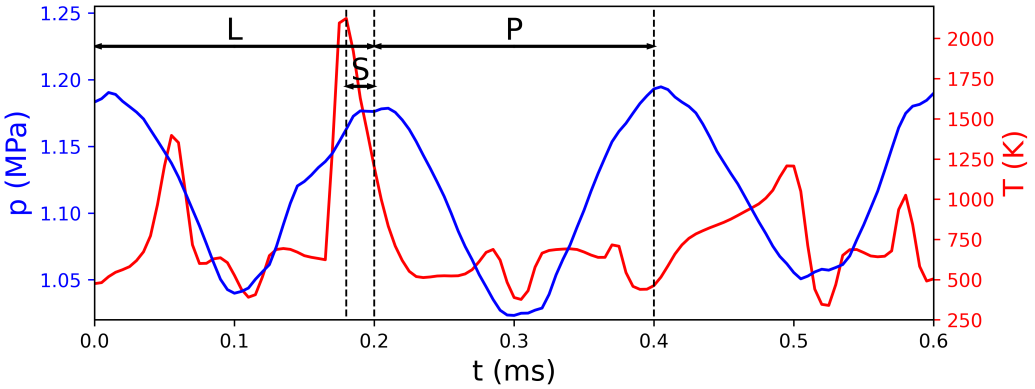
The injector is outlined in Fig. 4.7a. The oxidizer (O2 diluted in H2O vapor) and fuel (CH4) are injected from the large and the small inlets, respectively, into a tube-like combustion chamber, in which they mix and react. The products (CO2, H2O) are exhausted through the outlet. A probe monitor is placed inside the physics-intensive area, which is also marked in the figure. The strong instabilities in the simulation are triggered by a strong 2000 Hz pressure perturbation at the outlet. Fig. 4.7b shows the responses for p and T at the probe. It should be noted that, although the pressure perturbation at the outlet is periodic, the upstream behavior is affected by complex coupled physics and is not as periodic, especially for other variables such as T. Fig. 4.7c shows the graph generated using the cell-node mapping, where special nodes and edges, as well as irregular local structures, are provided in zoomed-in views. Two groups of ghost edges are used, corresponding to two types of wall boundary conditions in the simulation: no-slip and symmetry, respectively.

In this experiment, we attempt to predict the future states of $\mathbf{q}$ using the CP-GNet. Two CP-GNets of two different depths, with a 5-block and a 10-block processor respectively, are tested. Both CP-GNets work with an encoded node feature size of 36, and an encoded edge feature size of 4. The baseline model for comparison replaces all CP layers with standard dense layers of 128 units. More specifically, after the replacement, the layers taking node features as conditional parameters will retain the original input. The layers originally taking edge features as conditional parameters will take a concatenation of the original inputs and the edge features as the new input. The non-CP model is referred to as the GNet, for which a 10-block-processor and a 15-block-processor version are studied.

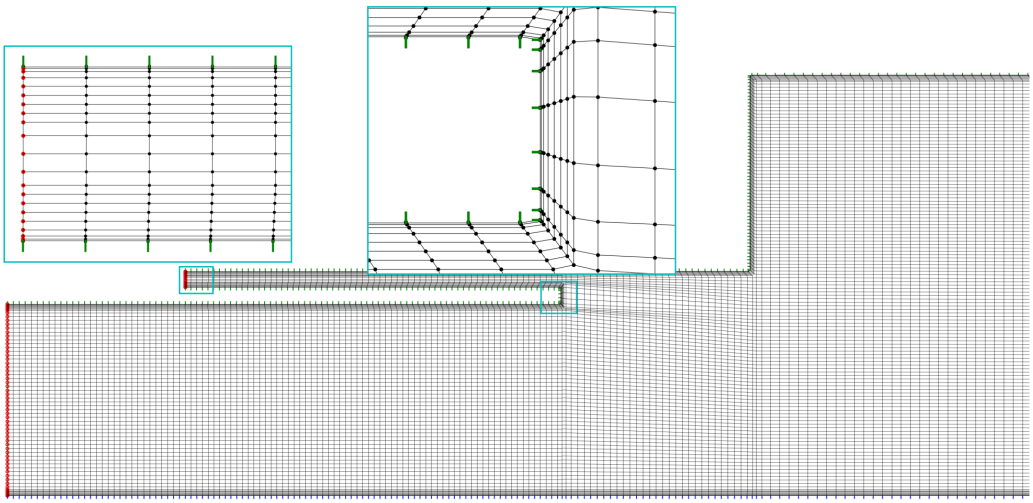
The original simulation is performed at a time-step size of 1×10^{-4} ms, and the results sampled at a time interval of 5×10^{-4} ms are used as the ground truth. Tests



(a) Injector outline. Orange marker shows the probe location.



(b) Probed response for p and T. L/S: long (0.2 s)/short (0.02 s) training period (0.2 s); P: prediction period (0.2 s).



(c) Graph details. Black dots: standard nodes; black lines: standard edges; red dots: inlet/outlet nodes; green/blue lines: two groups of ghost edges (extruded for visualization).

Figure 4.7: Illustrations for the reacting flow problem setup.

are conducted on two different lengths of training data. The long period consists of 400 steps, spanning 0.2 ms, the last 10% of which is used as the short training period. Thus, both periods end at the same point, and rollout prediction is carried out from the end of training for another 0.2 ms. These periods are illustrated in Fig. 4.7b. For simplicity, we add the number of processor blocks, and L (long) or

S (short) as suffixes to the model names to distinguish them. For example, "CP-GNet10L" refers to the CP-GNet with 10 processor blocks trained on the long period. In addition, the projection-reconstruction result from a 50-mode POD based on the long training period is also used for comparison. The number of modes is larger than those used in the related work (*McQuarrie et al.*, 2021; *Swischuk et al.*, 2020), and should reasonably represent the upper limit of any static POD-based result.

The probed results for all models and variables are plotted in Fig. 4.8. In addition, predicted flow fields for 4 representative variables, $p, u, T, Y_{\mathrm{CH4}}$, from the two deeper models, CP-GNet10L and GNet15L, are visualized in Fig. 4.9 at 4 steps evenly spanned over the prediction period, along with the POD reconstructions. It is notable that a small phase shift in the resolved structures can cause a high level of deviation in the probe measurements, and thus the flow field contours should be viewed as broader indicators of the performance. It is seen that the CP-GNET predicts the evolution of the reacting flow accurately over hundreds of prediction steps. In comparison, the non-CP model deviates quickly from the ground truth within 100 steps. Even with a smaller model (CP-GNet5L, 1.3M parameters), or a small fraction of training data (CP-GNet10S), the CP models still show comparable or even better performances compared with the largest baseline (GNet15L, 1.8M parameters). There is no significant difference in the level of error across the predicted field from our model, in spite of the vast changes in mesh density and distortion, whereas the GNets clearly suffer from more errors around the inner corners, where the mesh is the most irregular. This shows that, by combining CP with graph, discretization information can be efficiently processed. The POD reconstructions are even worse than the baseline GNet model in this case, except for the pressure, which behaves close to periodically due to the direct pressure perturbation. The advantage of local models over the global-projection-based ones for the advection of small-scale dynamics is again clearly demonstrated.

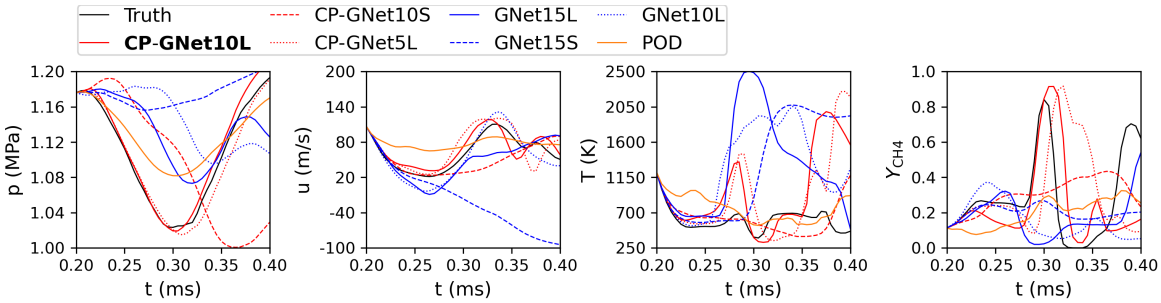


Figure 4.8: Probe history for 0.2 ms/400 snapshots following the end of training.

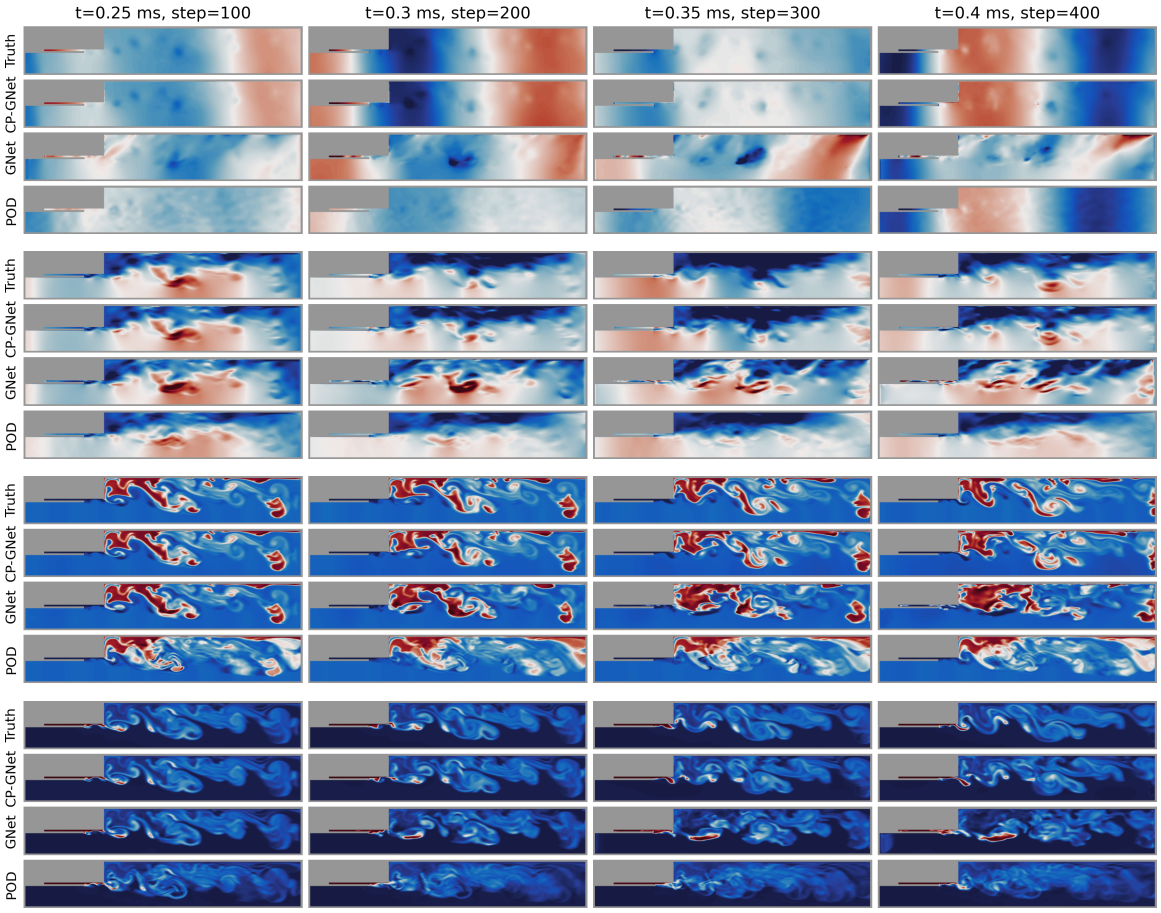


Figure 4.9: Predicted (CP-GNet10L, GNet15L) and reconstructed (POD) flow fields. From top to bottom: pressure p, x-velocity u, temperature T, mass fractions Y_{CH4}.

Results without the boundary treatment. The CP-GNet10L model is re-trained on a graph without ghost edges for the wall boundaries. The prediction is again started at the end of the training ($t = 0.2$ ms). The results for two representative variables, p and u, at the last prediction step $t = 0.4$ ms are shown in Fig. 4.10. The

predictions deviate noticeably further from the ground truth, and the accumulation of error is clearly visible in several near-wall regions, which validates our proposed boundary treatment in such a complex case with multiple types of boundaries.

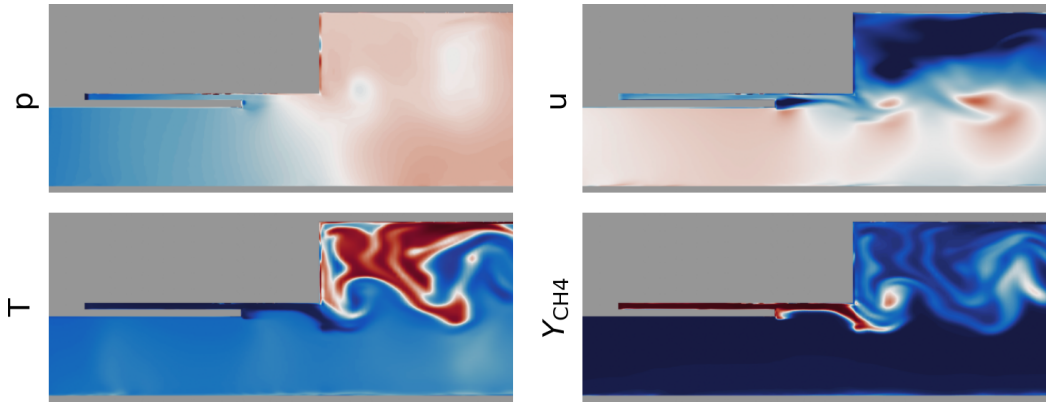


Figure 4.10: Zoomed-in view for results on a graph without ghost edges. The accumulation of error in the near-wall regions is clearly visible.

Prediction for a distanced and longer period. To test generalization performance, the CP-GNet10L model is used to perform prediction at a new time instance $t = 2$ ms, which is away from the training period. The predicted period is also doubled to 0.4 ms. The results are provided in Fig. 4.11 and 4.12, in comparison to the POD reconstructions. For the first 0.15 ms, a similar level of accuracy is obtained compared with the previous run that is appended to the end of the training period. However, the prediction is unstable in the long term, illustrated by scattered extreme values in the contours. Despite a significant improvement over the POD, which is again unable to capture meaningful dynamics, the author acknowledges that the lack of a long-time stability guarantee is still a major limitation of the current - and existing - work in the domain of data-driven flow predictions.

Timing. The prediction for 0.1 ms of flow with the CP-GNet10L model takes 53 seconds on one Nvidia RTX A6000 GPU, or 599 seconds on 40 CPU cores. In comparison, the original simulation for 6 ms of flow takes approximately 1200 CPU hours (*Mc-*

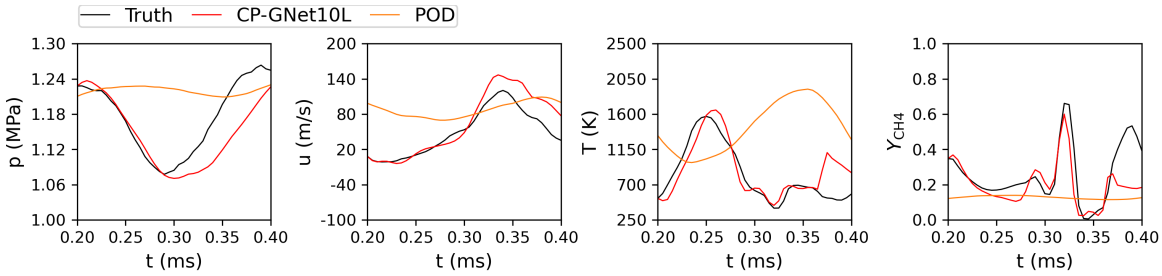


Figure 4.11: Probe history from a distanced time instance $t = 2$ ms, and for a longer period of 0.4 ms/800 snapshots.

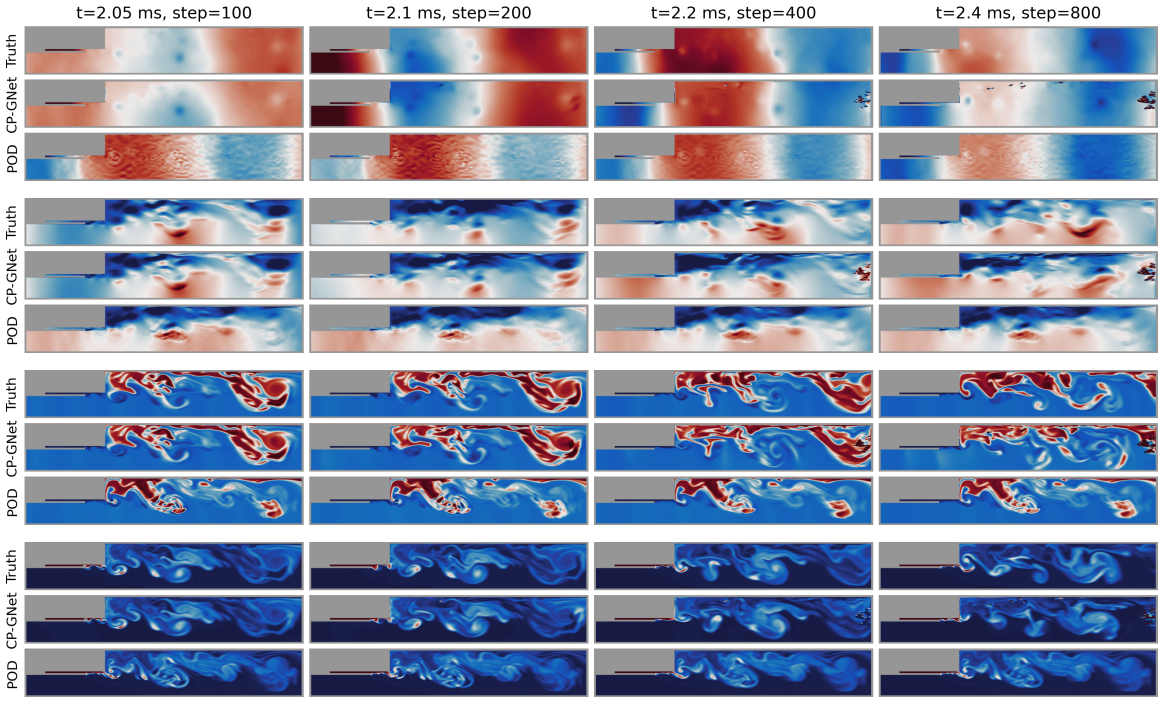


Figure 4.12: Predictions (CP-GNet10L)/reconstructions (POD) for the new IC and longer period. From top to bottom: pressure p, x-velocity u, temperature T, mass fractions Y_{CH4}. The selected time instances are not evenly spanned.

Quarrie et al., 2021). Due to different hardware configurations, no direct comparison can be made. However, we can safely estimate a 2.5x∼3x speedup on CPUs, and a 25x∼30x speedup when a GPU is utilized.

4.6.2 Incompressible Flow Over A Cylinder

For this experiment, the setting for incompressible flow over a cylinder from the MeshGraphNets (MGN) (*Pfaff et al.*, 2020) is adopted. The task is to predict the 2D

velocity components stored on the vertices on different irregular triangular meshes. The corresponding graphs are generated using the vertex-node mapping, with a 4-digit one-hot label added to the input node features to distinguish the 4 different node types: inner, inlet, outlet, and no-slip wall. The data includes 1000 training trajectories and 100 testing ones, each with 600 steps. Fig. 4.13 shows two sample meshes.

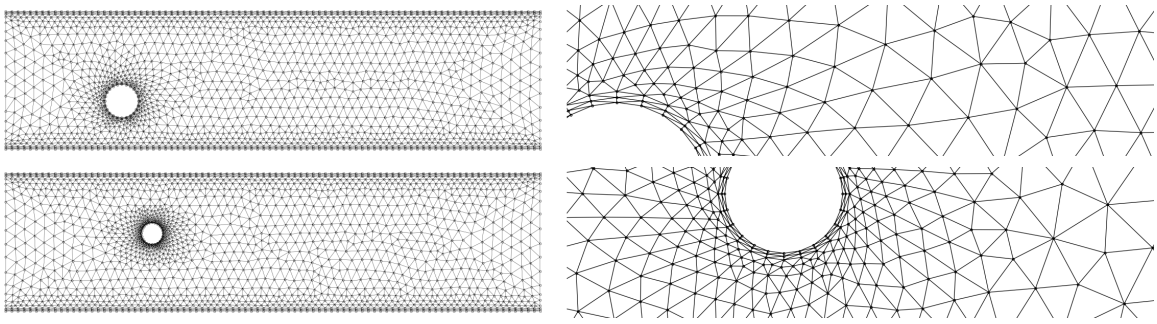


Figure 4.13: Two sample irregular meshes for the flow over a cylinder, with a zoomed-in view on the right.

The MGN results are generated using the official code, and compared against as the baseline. The MGN takes a 2-layer 128-unit MLP as the encoder for the node features, and an identical but independently trained one for the edge features. Each processor block also consists of two such MLPs with residual connections, f^e and f^u, for updating the edge and node latent features, respectively. The processor computation is formulated as:

$$\begin{aligned} \mathbf{e}'_{ij} &= f^e(\mathbf{e}_{ij}, \mathbf{u}_i, \mathbf{u}_j), \\ \mathbf{u}'_i &= f^u(\mathbf{u}_i, \sum_{j \in N(i)} \mathbf{e}'_{ij}). \end{aligned} \tag{4.14}$$

It can be seen that the MGN also concatenates the node and edge features. More complicated than the baseline GNet, a large (128-unit) edge feature is used, and gets updated through the processor blocks, with the node and edge features from the previous block. The node features are then updated in a following step in a similar

manner. Following the processors, another 2-layer MLP is used as the decoder that only works on the node features.

A 64-unit CP-GNet with 15 processor blocks is also migrated to the same pipeline. A two-layer MLP is added to the node encoder to process the node-type labels. An increase in the network width is made (from 36 in the previous test to 64) due to the additional processing of the additional features. The CP-Dense layer in the "source term" section in the CP-GNet processor is removed as there is no chemical reaction taking place in this test.

In the original setting, 10 million training steps are used, which takes several days on a single GPU. In this work, an additional comparison is performed after 2.5 million training steps to evaluate the training efficiency of the models. The averaged inference time and RMSE for the testing trajectories are summarized in Table 4.2. It should be noted that a single A6000 GPU is used in our tests, and a V100 is used by *Pfaff et al.* (2020), and noticeably different numbers are reported. At a smaller number of training steps (2.5 M), our model provides a higher single-step RMSE, yet a lower long-rollout RMSE. Compared with the previous test, the efficiency of our model is largely affected by the additional processing of node labels and a larger network width, running significantly slower than the MeshGraphNets. The predicted final steps for 5 randomly selected testing trajectories are visualized in Fig. 4.14. At 2.5 M training steps, the CP-GNet performs better in two unsteady cases (trajectories #8 and #17). However, it over-predicts the velocity magnitude in the two steady cases (trajectories #32 and #72). Meanwhile, the MeshGraphNet under-predicts in one of them (trajectory #72). At the larger number of training steps (10 M), the MeshGraphNets shows a lower error across the prediction period, with the gap between the two models decreasing with the number of rollout steps. The differences in the final step predictions become less significant.

The MGN is also applied to the previous reacting flow prediction task, where

Table 4.2: Averaged inference time and RMSE for flow over a cylinder.

Model (training steps)	Time/step ms	RMSE 1-step $\times 10^{-3}$	RMSE rollout-50 $\times 10^{-3}$	RMSE rollout-all $\times 10^{-3}$
CP-GNet (2.5 M)	16	3.3	12.4	62.5
CP-GNet (10 M)	16	2.8	9.9	54.0
MGN (2.5 M, tested)	9	2.1	8.7	68.5
MGN (10 M, tested)	9	1.9	6.9	50.1
MGN (10 M, reported)	21	2.34 ± 0.12	6.3 ± 0.7	40.88 ± 7.2

additional node-type labels are added to maintain the original MGN architecture in the cell-node mapping. The results are provided in Appendix B.1, where the CP-GNet outperforms the MGN. It should be pointed out that between these problems, there are many factors that can lead to changes in model performances, including the type of the ground truth solver and data (cell-centered FVM vs. vertex-centered FEM), the number of variables (8 vs. 2), the implement of boundary conditions (ghost edges vs. node labels). Based on these specific results, one cannot make a definitive statement on the relative merits of the MGN (updating a wide set of edge features through each processing block) and the CP-GNet (parameterizing on a narrow and fixed set of edge features.).

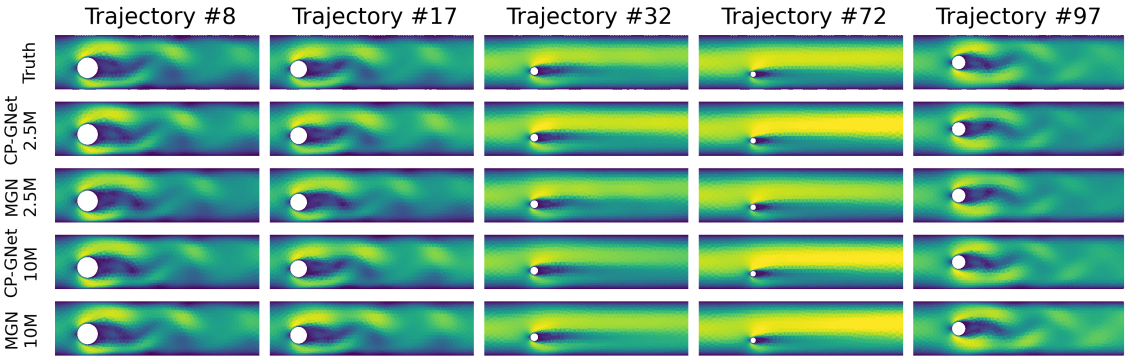


Figure 4.14: Velocity magnitude for the last step in the rollout prediction for random testing trajectories. From top to bottom: ground truth, CP-GNet, MGN.

4.7 Summary

This chapter starts with an introduction to the idea of CP and its implementation in neural networks. The advantage of a local CP model is clearly demonstrated by fitting a discretized 2D advection-diffusion equation exactly with less than 10 snapshots, which is not possible with projection-based ROM even when the same equations are used intrusively.

After the idealized demonstration, the mesh-to-graph mapping procedures for different spatial discretion methods are then described formally, which is followed by the introduction of the CP-GNet. The CP-GNet draws inspiration from discretized numerical methods, and generalizes the idea of CP for mesh-based models, which enables a flexible incorporation of physical quantities as well as numerical discretization information into trainable weights, leading to efficient learning of high-order and unstructured features. In a test of future state prediction of a rocket injector that is closely related to the target application of our multi-fidelity framework, the CP-GNet is shown to be capable of predicting the flow with complex combustion processes for a few hundred steps on an irregular mesh. In another test case, the CP-GNet is shown to be able to predict transient flow over a cylinder in unseen geometries/meshes, and trains more efficiently than the MeshGraphNet. Although a direct CP modification will cause a linear increase in the number of parameters w.r.t. the chosen parameter, such an increase can be compensated by reducing the size of the latent vectors. Indeed, the CP-GNet is more efficient than the non-CP variant with only a fraction of the training data or with a more shallow architecture.

In Appendix A, drop-in CP modifications are demonstrated on different architectures for two other important tasks related to mesh-based modeling of physical systems, coarse-graining, and super-resolution. Considerable performance improvements are achieved compared with the traditional counterparts. Overall, the proposed architecture improves the potential for incorporating physical intuition as well

as knowledge of numerical discretization, and serves as a promising model for our framework.

CHAPTER V

Coupling a Reduced-Complexity Model with a High-Fidelity Solver

5.1 Introduction

The core merit of the overall framework of this thesis is to exploit the efficiency of a reduced-complexity model and the flexibility of a high-fidelity solver simultaneously. Besides the standalone development of RCMs, an important but largely under-explored aspect of this strategy is the coupling method between different types of models and solvers. In this chapter, we first provide an overview of the model-solver coupling procedures in a decomposed domain, with a few definitions that are used throughout the rest of the chapter, which is further explained with the relatively straightforward intrusive coupling between physical variables. Two major challenges are then considered: 1) effective communication with non-intrusive models where the interface conditions cannot be directly processed by governing equations; and 2) coupling between mismatched time-integration schemes.

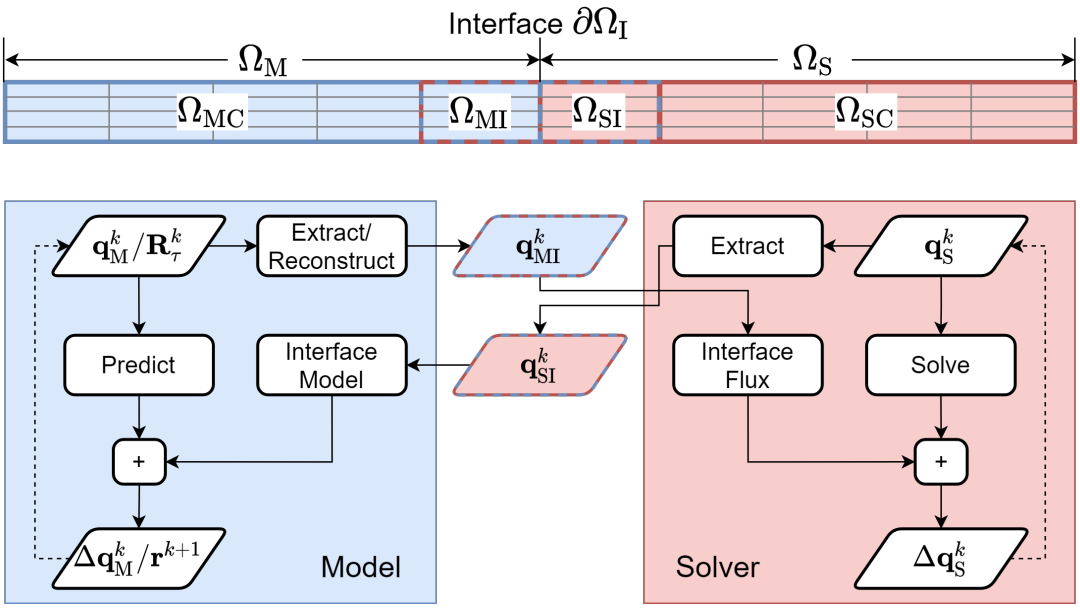


Figure 5.1: Definitions in domain decomposition (top), and model-solver coupling procedures (bottom).

5.2 Overview of Domain Decomposition and Coupling Procedures

Fig. 5.1 shows a few definitions for different parts in a decomposed domain, and overall procedures in model-solver coupling. The computational domain is assumed to be decomposed into a RCM sub-domain Ω_M, and a solver sub-domain Ω_S. The layer of cells immediately adjacent to the interface $\partial\Omega_I$ is marked as Ω_{MI} and Ω_{SI} on the two sides of the interface, respectively. The "self-contained" parts are denoted as Ω_{MC} and Ω_{SC}, respectively. In the coupled online prediction, the values in the interfaces layers $\mathbf{q}_{MI}^k = \{\mathbf{q}_i^k; n_i \in \Omega_{MI}\}$ and $\mathbf{q}_{SI}^k = \{\mathbf{q}_i^k; n_i \in \Omega_{SI}\}$ are sent to the opposite sub-domains. For demonstration purposes, the schematic assumes that communications take place at the beginning of a time-step k. As will be demonstrated in Sec. 5.4, communications may also be performed at a different interval when mismatched schemes are used.

5.2.1 Coupling Intrusive Models

In a FOM Eq. (1.2) or an intrusive POD-Galerkin ROM Eq. (1.6), the "external" interface values, i.e. the received $\mathbf{q}_{\mathrm{MI}}^k/\mathbf{q}_{\mathrm{SI}}^k$, are directly used in the computation in the non-linear term $\mathbf{f}$ for the "internal" interface values $\mathbf{q}_{\mathrm{SI}}^k/\mathbf{q}_{\mathrm{MI}}^k$ as their neighboring cells. This is similar to communicating different partitions in a parallel computation, which can usually be accomplished with existing routines, and have been verified in related work (*Huang et al.*, 2016, 2017; *Xu et al.*, 2019). One observation worth pointing out is that the dependency on neighboring cells for $\mathbf{f}$ only lies in the computation for the spatial derivative, e.g. the flux $\hat{\mathbf{F}}$ in Eq. (4.8), which is important for the development of the interface models for the non-intrusive coupling in the next section.

5.2.2 Coupling with the CP-GNet

The coupling with the CP-GNet is similar to the intrusive coupling, because the model also directly works on the physical quantities. To receive the external interface values, the graph G should include both Ω_{M} and Ω_{SI}. In the online prediction, the values on the graph nodes corresponding to the mesh for Ω_{SI} are updated with the received values from the coupled solver, instead of using the graph-based model[1].

5.3 Coupling with Reduced Order Variables

Due to an inconsistency in the variables used, the coupling with a non-intrusive POD ROM requires additional treatments. First, the values to send to the other side of the interface need to be reconstructed from the reduced variables as:

$$\mathbf{q}_{\mathrm{MI}}^k = \mathbf{V}_{\mathrm{MI}}\mathbf{r}^k, \tag{5.1}$$

[1] A graph-based model such as the CP-GNet will still predict for these nodes, but the predictions are ignored.

where $\mathbf{V}_{\mathrm{MI}}$ denotes a POD basis for $\mathbf{q}_{\mathrm{MI}}$.

Secondly, an interface model is needed to incorporate the received physical variables $\mathbf{q}_{\mathrm{SI}}^k$ with the existing model Eq. (3.9). In this work, three types of interface models are realized.

5.3.1 Strategy 1: Flux Projection

It can be realized that the prediction of $\mathbf{r}^{k+1}$ and $\Delta\mathbf{r}^k$ is interchangeable. Assuming a forward-Euler time-integration scheme, which is common for RCMs, the ideal prediction for the latter equals a linear combination:

$$\Delta\mathbf{r}^k = -\Delta t \sum_{n_i \in \Omega_{\mathrm{M}}} \mathbf{V}_{i,:}^T \mathbf{f}_i^k. \tag{5.2}$$

In a FVM-based system Eq. (4.9), $\mathbf{f}_i^k$ is again a linear combination of the fluxes on all cell edges/faces, and the source term. As pointed out in the previous section, external interface cells are only involved in the fluxes for interface edges/faces. Therefore, Eq. (5.2) can be further decomposed as:

$$\begin{aligned} \Delta\mathbf{r}^k &= \Delta\mathbf{r}_{\mathrm{M}}^k + \Delta\mathbf{r}_{\mathrm{I}}^k, \\ \Delta\mathbf{r}_{\mathrm{M}}^k &= -\Delta t \left[\sum_{n_i \in \Omega_{\mathrm{MC}}} \mathbf{V}_{i,:}^T \mathbf{f}_i^k + \sum_{e_{ij} \notin \partial\Omega_{\mathrm{I}}} \frac{L_{ij}}{A_i} \mathbf{V}_{i,:}^T \hat{\mathbf{F}}_{ij}^k - \sum_{n_i \in \Omega_{\mathrm{MI}}} \mathbf{V}_{i,:}^T \mathbf{S}_i^k \right], \\ \Delta\mathbf{r}_{\mathrm{I}}^k &= -\Delta t \sum_{e_{ij} \in \partial\Omega_{\mathrm{I}}} \frac{L_{ij}}{A_i} \mathbf{V}_{i,:}^T \hat{\mathbf{F}}_{ij}^k, \end{aligned} \tag{5.3}$$

where $\Delta\mathbf{r}_{\mathrm{M}}^k$ denotes the terms that only depends on $\mathbf{q}_{\mathrm{M}}^k$, and $\Delta\mathbf{r}_{\mathrm{I}}^k$ denotes those depending on both $\mathbf{q}_{\mathrm{MI}}^k$ and $\mathbf{q}_{\mathrm{SI}}^k$. Because $\mathbf{R}_\tau$ and $\mathbf{V}_{\mathrm{M}}$ only stores sufficient information for the former, the non-intrusive model Eq. (3.9) should not be expected to predict for both terms when coupled with a solver. Therefore, it modified as:

$$\mathcal{F}_{\mathrm{M}}(\mathbf{R}_\tau^k) = \mathbf{r}^k + \Delta\mathbf{r}_M^k. \tag{5.4}$$

After the modification, the coupled online prediction is given by:

$$\mathbf{r}^{k+1} = \mathcal{F}_{\mathrm{M}}(\mathbf{R}_\tau^k) + \Delta\mathbf{r}_{\mathrm{I}}^k. \tag{5.5}$$

5.3.2 Strategy 2: Interface MLP

In the previous method, the interface model is a direct projection of the interface flux onto the basis for the internal interface cells, and the autoregressive model works independently on the complementary part in the increment $\Delta\mathbf{r}$. One problem with this approach is that $\Delta\mathbf{r}$ is often zero-centered statistically, whereas $\Delta\mathbf{r}_{\mathrm{I}}$ is not, because the mean of the fluxes is usually non-zero, determined by the flow direction. [2] Therefore the complementary part $\Delta\mathbf{r}_{\mathrm{M}}$, which is the target output for $\mathcal{F}$, is also not zero-centered. In long-term prediction, this makes the accumulative error more significant, due to the existence of a possibly large non-zero bias. Even when the target output is carefully scaled, the influence of such a bias cannot be eliminated, because in the online prediction the output has to be re-scaled and the bias is added back.

One way to address the problem is to design an interface model that works with $\mathcal{F}$, such that both models predict for $\Delta\mathbf{r}$ jointly instead of independently. In our work, a MLP model $\mathcal{F}_{\mathrm{I}}(\mathbf{q}_i, \mathbf{q}_j)$ is used, which takes the values for both the internal and the external interface cells as the input. The coupled prediction is then given by:

$$\mathbf{r}^{k+1} = \mathcal{F}_{\mathrm{M}}(\mathbf{R}_\tau^k) + \sum_{e_{ij}\in\partial\Omega_{\mathrm{I}}} \mathcal{F}_{\mathrm{I}}(\mathbf{q}_i^k, \mathbf{q}_j^k). \tag{5.6}$$

It should be noted that although $\Delta\mathbf{r}_{\mathrm{I}}$ is assumed to be the summed output for $\mathcal{F}_{\mathrm{I}}$ by concept, this relationship does not present explicitly in the training process for Eq. (5.6). Instead, the two models $\mathcal{F}_{\mathrm{M}}$ and $\mathcal{F}_{\mathrm{I}}$ are trained jointly for one target

[2] There are exceptions such as when two interfaces are present at the two ends of a conservative duct, so that their fluxes compensate for each other. We focus our discussion on more general cases.

output $\Delta\mathbf{q}$, which solves the problem described at the beginning of this section.

5.3.3 Strategy 3: Integration Through an Overlapped Projection

The shared target for all coupling methods in this section is to use the information on Ω_{M} and Ω_{SI}, and predict for the reduced order variable $\mathbf{r}$. If we make the domain decomposition slightly more flexible, and take Ω_{SI} as an overlapped "buffer" layer such that $\mathbf{q}_{\mathrm{SI}}$ is stored in the high order space in the solver, and at the same time included in an extended projection basis $\mathbf{V}_{\mathrm{M}\cup\mathrm{SI}}$ in the solver, then the corresponding reduced order variable is computed by a projection:

$$\mathbf{r} = \mathbf{V}_{\mathrm{M}\cup\mathrm{SI}}^T \begin{bmatrix} \mathbf{q}_{\mathrm{M}} \\ \mathbf{q}_{\mathrm{SI}} \end{bmatrix} = [\mathbf{V}_{\mathrm{M}}^T, \mathbf{V}_{\mathrm{SI}}^T] \begin{bmatrix} \mathbf{q}_{\mathrm{M}} \\ \mathbf{q}_{\mathrm{SI}} \end{bmatrix} = \mathbf{V}_{\mathrm{M}}^T \mathbf{V}_{\mathrm{M}} \mathbf{r} + \mathbf{V}_{\mathrm{SI}}^T \mathbf{q}_{\mathrm{SI}}. \tag{5.7}$$

In the expression above, the influence of the external interface layer on the reduced order variable is integrated in the projection process. If we build an autoregressive model for the first term as $\mathcal{F}_{\mathrm{M}}(\mathbf{R}_\tau^k) = \mathbf{V}_{\mathrm{M}}^T \mathbf{V}_{\mathrm{M}} \mathbf{r}^{k+1}$, then a coupled model is given by:

$$\mathbf{r}^{k+1} = \mathcal{F}_{\mathrm{M}}(\mathbf{R}_\tau^k) + \mathbf{V}_{\mathrm{SI}}^T \mathbf{q}_{\mathrm{SI}}, \tag{5.8}$$

5.3.4 Comparison Between Interface Models

The wave propagation problem from Sec. 3.4, and the POD-MLP, POD-LSTM models are used to compare the different reduced order interface models. The continuity in the test case helps to illustrate the importance of an effective interface coupling strategy. To focus on the influence of the coupling, and increase the significance of the interface models, the 1D domain is truncated into three sub-domains, following a FOM-ROM-FOM arrangement, as shown in Fig. 5.2. The ROM part is made short, including only 20 cells (leaving 90 cells in each FOM sub-domain), such that the ratio of $\Delta\mathbf{r}_{\mathrm{I}}$ over $\Delta\mathbf{r}_{\mathrm{M}}$ is larger. When a trainable interface model, i.e. the

interface MLP, is used, two identical but independently trained models are used for the two interfaces. Each MLP is similar to the one used as the autoregressive model, with the input size increased from n_r to $2n_v$ (for variables on both sides of the interface). When combined with different interface models, the autoregressive models are also identical but independently trained. For additional reference, a coupled intrusive POD-Galerkin ROM as described in Sec. 5.2 is also tested.

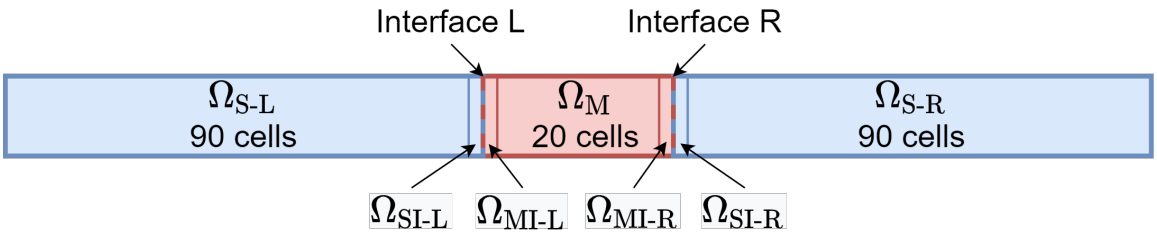


Figure 5.2: FOM-ROM-FOM domain decomposition.

To examine the effectiveness of the coupling method, the response at the four interface cells, $\Omega_{\text{SI-L}}, \Omega_{\text{MI-L}}, \Omega_{\text{MI-R}}, \Omega_{\text{SI-R}}$, are plotted in Fig. 5.3. In supersonic flow, the response curve for the downstream cells should follow the upstream ones with a phase difference proportional to their distance. Marked in different colors, the curves fall into two pairs, with the left pair, $\Omega_{\text{SI-L}}$ and $\Omega_{\text{MI-L}}$, reflecting the capability of a model processing information from a solver, and the right pair, $\Omega_{\text{MI-R}}$ and $\Omega_{\text{SI-R}}$, reflecting a solver processing information from a model. The RMSE for the coupled solution is also reported in the same figure.

Similar to the full-domain model behaviors in Sec. 3.4, when no interface model is used ("standalone"), the MLP is unable to predict the change in the dynamic frequency, which is well learned by the LSTM. When interface models are applied, however, the relevant performance between the two models becomes different. The flux projection ("flux") model enables the MLP to accurately capture the upstream change of frequency, resulting in the lowest RMSE among all non-intrusive models. However, the "LSTM flux" combination performs even worse than the standalone

LSTM model. This is caused by a change of target output to $\Delta\mathbf{r}^k - \Delta\mathbf{r}_{\mathrm{I}}^k$, from the original $\Delta\mathbf{r}^k$ that is in favor of the RNN type of autoregression based on a long look-back window for $\mathbf{r}$, instead of $\mathbf{r} - \mathbf{r}_{\mathrm{I}}$. In comparison, the MLP interface model is able to improve both original models significantly, demonstrating the effectiveness of the use of joint optimization in the model training. The overlapped projection model ("overlapped") however, is only able to correct the phase of the MLP prediction, but causes both models to predict wrong overall dynamics. Based on the results, a jointly optimized trainable boundary model is a more favorable one for the non-intrusive models in this study.

5.4 Coupling Between Mismatched Time-Integration Schemes

The discussions in this chapter so far have assumed a shared time-integration scheme for both the solver and the RCM. As introduced in Sec. 1.4.2, it is not straightforward to couple a solver with a model at a coarser scheme. Taking a FVM solver for example, we assume that Ω_i is an internal cell that is time-integrated using a multi-stage scheme, e.g. the Runge Kutta method. We also assume that Ω_j is an external cell adjacent to Ω_i, but time-integrated using a forward-Euler-like scheme that updates only once every time-step, corresponding to the last stage for Ω_i. At the first stage for Ω_i, the flux between the two cells is computed in a conservative way, i.e. the amount of conserved variables going out from Ω_i (can be negative) through the interface should be equal to that going into Ω_j from the interface. However, because that only $\mathbf{q}_i$ gets updated by the flux after the first stage and $\mathbf{q}_j$ does not, the conservation assumption is broken. Actually, except for the last stage, the fixed $\mathbf{q}_j$ can be viewed as a Dirichlet boundary condition imposed on $\mathbf{q}_i$, which will lead to problems such as reflections of waves.

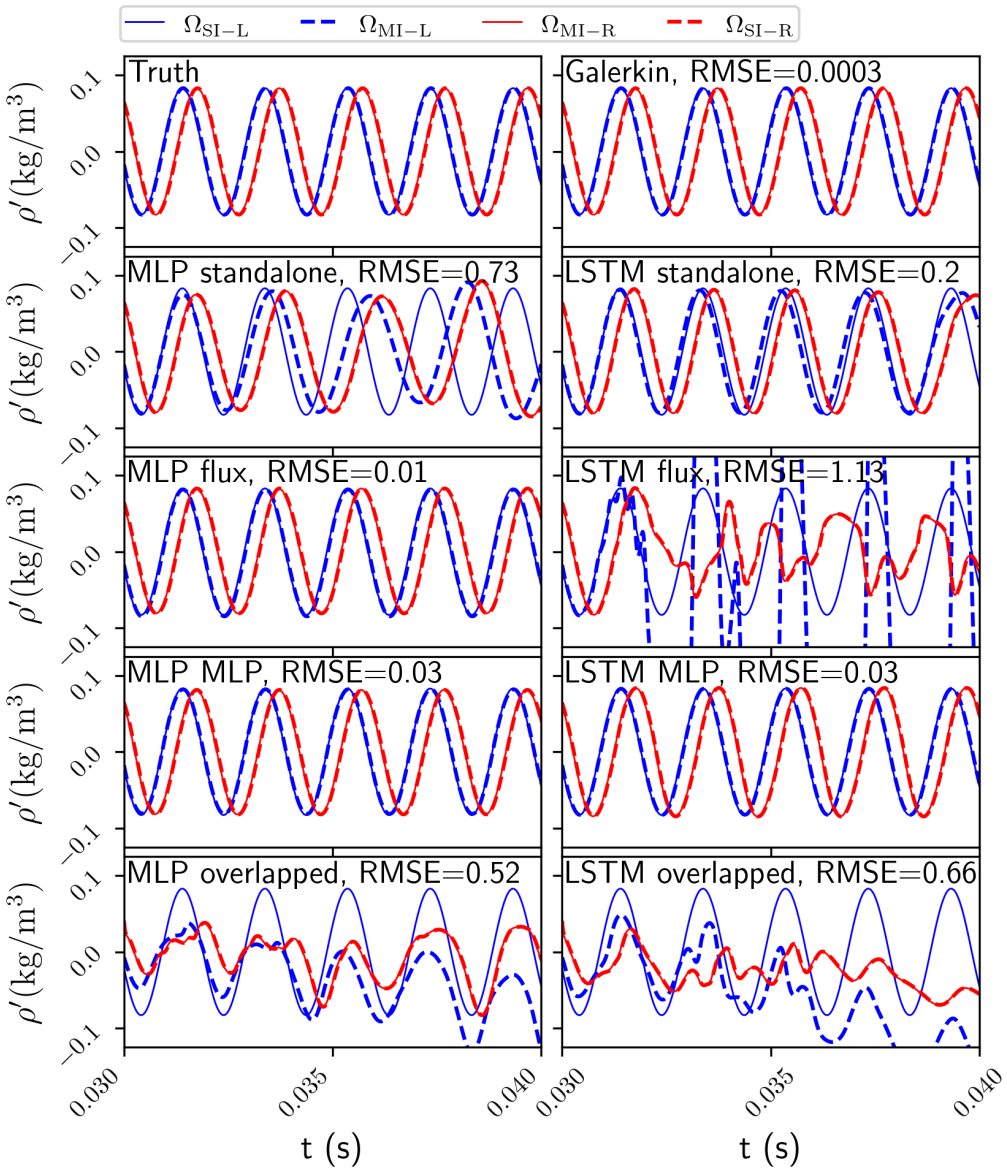
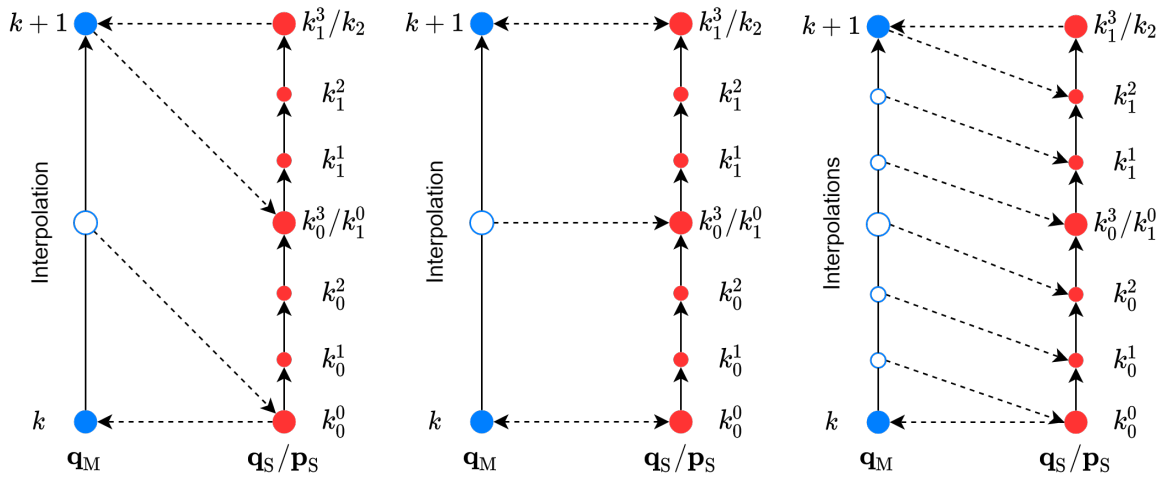


Figure 5.3: Interface cell responses and the RMSE for the coupled solution. Each color represents a pair of cells around a different interface. The titles consist of the autoregressive model name followed by the interface model name, i.e. the "LSTM MLP" denotes a LSTM autoregressive model combined with an interface MLP model.

5.4.1 Comparison of Staggered Schemes

There exist multiple flavors of staggered schemes that may be applicable to alleviate the mismatching problem. However, their effectiveness remains to be evaluated in a model-solver coupling setting. In this sub-section, two of the most popular stag-

gered schemes for Fluid-Structure Iterations (FSI), the Conventional Serial Staggered (CSS), and the Conventional Parallel Staggered (CPS) schemes are introduced. The introductions are in a specialized setting for the purpose of this work, in which we assume: 1) that the time-step size of a RCM Δt_{M} is a n_{I} multiple of that of a solver Δt_{S}, and 2) that no sub-iterations are used by the RCM, and multiple ones are used by the solver. For the more general introduction of the schemes, mostly in FSI, readers are referred to *Farhat and Lesoinne* (2000) and *Gatzhammer* (2014). Subsequently, a development of the CSS specialized for the coupling between a RCM and a multi-stage solver is proposed, which applies additional temporal interpolation to the RCM solution, corresponding to the sub-iterations of the solver. The resulting scheme is named the Sub-Iteration Serial Staggered (SISS) scheme. In the presentation of the following algorithms, the main loop of time-integration is based on the model time-step k from 0 (IC) to n_t. Within k, the finer solver steps are denoted as $k_0, \ldots, k_{n_{\mathrm{I}}}$, where k_0 matches k, k_1 leads k by Δt_{S}, and $k_{n_{\mathrm{I}}}$ matches $k+1$. For sub-stages of the finer step, we follow the nomenclature used in Sec. 4.5.1.2, use $\mathbf{p}$ for the sub-stage solution up to P total sub-stages, and use k_i^p to denote the p-th sub-stage in the k_i-th finer step. Fig. 5.4 provides a direct comparison of the schemes for $n_{\mathrm{I}} = 2, P = 3$.



(a) CSS (b) CPS (c) SISS

Figure 5.4: Comparison of staggered schemes for $n_{\mathrm{I}} = 2, P = 3$.

CSS. In CSS, one of the coupled models/solvers is time-integrated first, and the updated solution is used for the time-integration of the other one. In our setting, the CSS procedure is given in Algorithm. 1, in which the RCM is first advanced from k to $k+1$. For $n_{\mathrm{I}} > 1$, linear interpolation is performed for $\mathbf{q}_{\mathrm{MI}}$ at k_{i+1}. the interpolated result is used to advance $\mathbf{q}_{\mathrm{S}}$ from k_i to k_{i+1}.

Algorithm 1 CSS procedure

1: **for** k=0 to $n_t - 1$ **do**
2: compute $\mathbf{q}_{\mathrm{M}}^{k+1}$ in RCM
3: **for** i=0 to $n_{\mathrm{I}} - 1$ **do**
4: **if** $n_{\mathrm{I}} > 1$ **then**
5: perform linear interpolation of $\mathbf{q}_{\mathrm{MI}}$ to k_{i+1}.
6: **end if**
7: send $\mathbf{q}_{\mathrm{MI}}^{k_{i+1}}$ to solver
8: compute $\mathbf{q}_{\mathrm{S}}^{k_{i+1}}$ in solver {Sub-iterations omitted in the absence of interface communication}
9: **end for**
10: send $\mathbf{q}_{\mathrm{SI}}^{k+1}$ to RCM
11: **end for**

CPS. The CPS procedure is given in Algorithm. 2. The major difference from the CSS is that the interpolated model solution for k_i, instead of k_{i+1}, is used for the time-integration of the solver from k_i to k_{i+1}. The scheme is regarded as "parallel" because when the same time-step sizes are used, i.e. $n_{\mathrm{I}} = 1$, the computations for $\mathbf{q}_{\mathrm{M}}^{k+1}$ and $\mathbf{q}_{\mathrm{S}}^{k+1}$ are both based on the old solution of their partners. Therefore, the computation can be performed in parallel. When mismatched time-steps are used, such parallelism is limited to the computation of the first finer step. In *Farhat and Lesoinne* (2000), the CPS is shown to be less accurate than the CSS, and only recommended when parallelism is necessary.

SISS. Neither the CSS nor the CPS perform interpolation and communication at the sub-stages of the solver. Therefore, the problem due to a fixed interface value could still take place for $P > 1$. The SISS procedure is given in Algorithm. 3, which

Algorithm 2 CPS procedure

1: **for** k=0 to $n_t - 1$ **do**
2: compute $\mathbf{q}_{\mathrm{M}}^{k+1}$ in RCM
3: compute $\mathbf{q}_{\mathrm{S}}^{k_1}$ in solver
4: **if** $n_{\mathrm{I}} > 1$ **then**
5: **for** i=1 to $n_{\mathrm{I}} - 1$ **do**
6: perform linear interpolation of $\mathbf{q}_{\mathrm{MI}}$ to k_i.
7: transfer $\mathbf{q}_{\mathrm{MI}}^{k_i}$ to solver
8: compute $\mathbf{q}_{\mathrm{S}}^{k_{i+1}}$ in solver {Sub-iterations omitted in the absence of interface communication}
9: **end for**
10: **end if**
11: send $\mathbf{q}_{\mathrm{MI}}^{k+1}$ to solver
12: send $\mathbf{q}_{\mathrm{SI}}^{k+1}$ to RCM
13: **end for**

can be viewed as an extension of the CSS to the sub-stage level. It should be pointed out that, although proposed in the present model-solver coupling context for the first time, the idea behind SISS, to perform communication at the sub-stages, is not entirely novel. Similar approaches have been taken for the coupling of two high-order schemes, leading to the High Order Implicit-Explicit (IMEX) schemes (*van Zuijlen and Bijl*, 2004; *Van Zuijlen et al.*, 2007).

Algorithm 3 SISS procedure

1: **for** k=0 to $n_t - 1$ **do**
2: compute $\mathbf{q}_{\mathrm{M}}^{k+1}$ in RCM
3: **for** i=0 to $n_{\mathrm{I}} - 1$ **do**
4: **for** p=0 to $P - 1$ **do**
5: perform linear interpolation of $\mathbf{q}_{\mathrm{MI}}$ to k_i^{p+1}.
6: send $\mathbf{q}_{\mathrm{MI}}^{k_i^{p+1}}$ to solver
7: compute $\mathbf{q}_{\mathrm{S}}^{k_i^{p+1}}$ in solver
8: **end for**
9: **end for**
10: send $\mathbf{q}_{\mathrm{SI}}^{k+1}$ to RCM
11: **end for**

5.4.2 Numerical Tests on Subsonic Inviscid Flow

To demonstrate the aforementioned problem, and to examine the effectiveness of different staggered schemes, subsonic inviscid flow is used. Fig. 5.5 shows the geometry and different sub-domains, and the problem setup is summarized in Table 5.1. It should be pointed out that the flow is span-wise uniform, despite the 2D computational domain. A sinusoidal perturbation at a relative amplitude of 10% is added to the outlet back-pressure, which triggers unstable dynamics represented by upstream-running waves. Fig. 5.6 shows an envelope comprised of the pressure profiles along the streamwise centerline for a complete cycle spanning 50 time-steps. The problem is solved with the GEMS, with a DTS scheme (Eq. (4.10)) with $P = 20$. It should be reminded that the initial stage $\mathbf{p}^0 = \mathbf{q}^k$, and the final stage $\mathbf{p}^P$ converges to $\mathbf{q}^{k+1}$. Any stage between 0 and P does not match with either $\mathbf{q}^k$ or $\mathbf{q}^{k+1}$, and the aforementioned problem will take place, and a consequential mismatch will be reflected by a steep gradient across the interface.

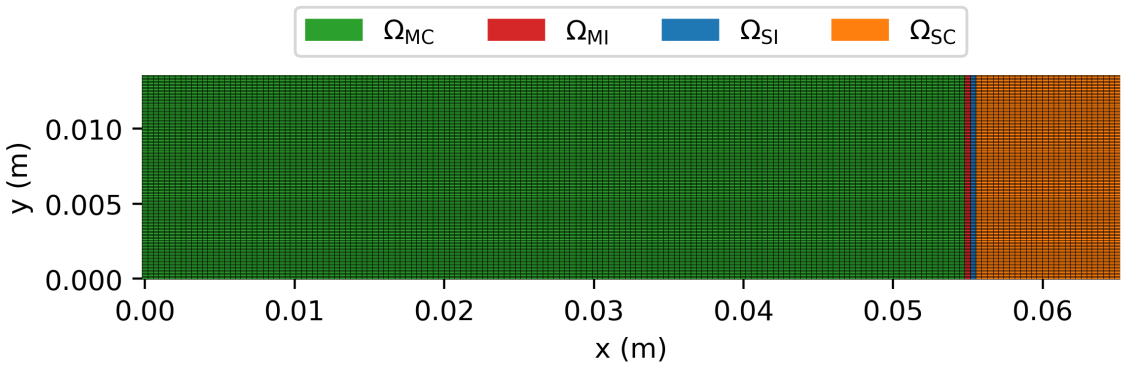


Figure 5.5: Geometry and sub-domains for 2D channel flow.

5.4.2.1 Truth-Solver Coupling

To eliminate the potential influence of an inaccurate model, and to narrow down the source of error to the time-integration mismatch only, we first couple GEMS with a "ground truth feeder" instead of a real model. The truth to be injected is pre-

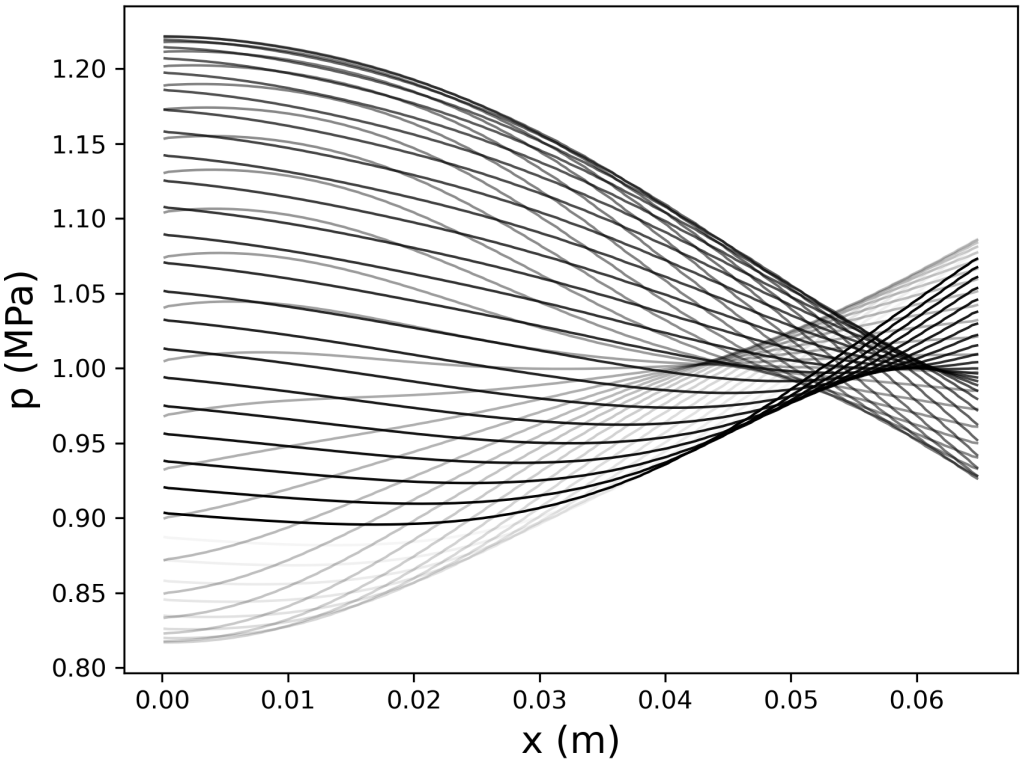


Figure 5.6: Pressure profile envelope comprised of 50 steps.

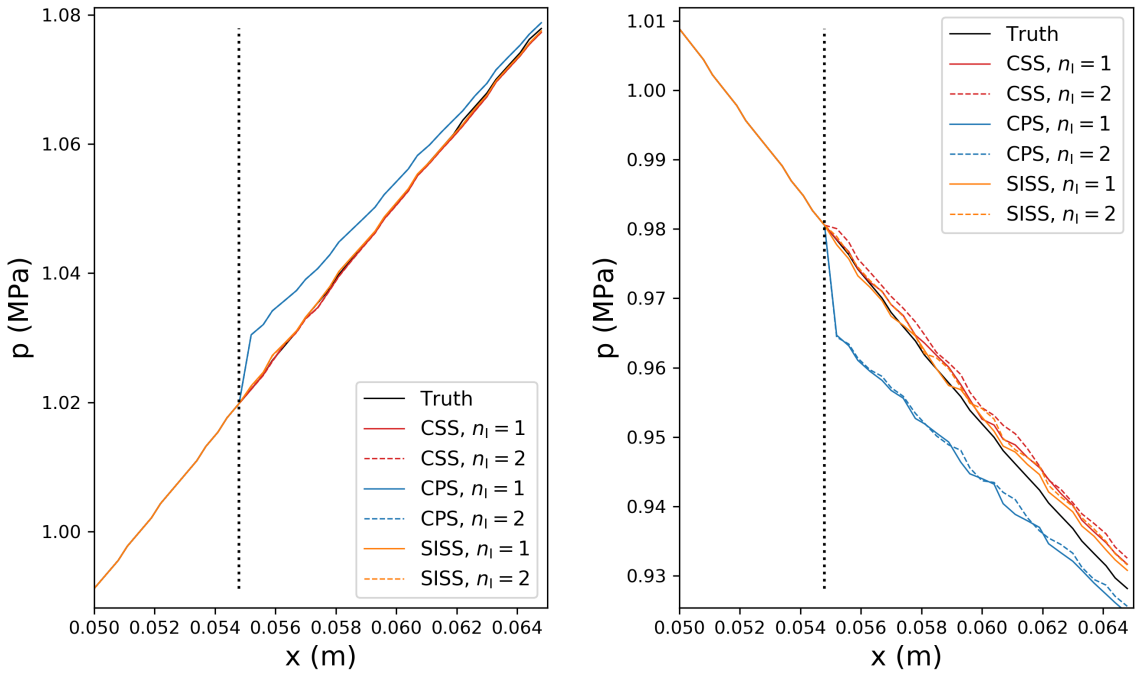
generated using GEMS for the full domain. Therefore, without any error introduced by the coupling, the feeder can be viewed as an exact model. This truth-solver coupling is single-directional – at the beginning of each time-step, the feeder sends the ground truth for $\mathbf{q}_{\mathrm{MI}}^{k}$ to GEMS, regardless of what GEMS predicted in the previous step. Two coarse-to-fine ratio, $n_{\mathrm{I}} = 1$ and $n_{\mathrm{I}} = 2$ are tested. In either case, the solver time-step size is fixed, and the model time-step size, i.e. the frequency that the GT is fed, is adjusted.

The results are visualized Fig. 5.7 for two fine time-steps, $k_{\mathrm{S}} = 1$ and $k_{\mathrm{S}} = 25$. For the CPS, the error across the interface is severe even at the first step, and increases with k_{S}. In comparison, the CSS reduces the error significantly. For $n_{\mathrm{I}} = 1$, the profile for the CSS stays close to the truth at the interface, and deviates linearly with an increased distance. However, for $n_{\mathrm{I}} = 2$, a steep gradient is observed at the interface.

Table 5.1: Problem setup.

Parameter	Value
Geometry length×height	0.065m × 0.0135m
Inflow velocity	200 m/s
Inflow temperature	2400 K
Inflow pressure	1.1 MPa
Mean outflow back-pressure	0.8 MPa
Back-pressure perturbation amplitude	0.08 MPa (10%)
Back-pressure perturbation frequency	2000 Hz
Time-step size	1×10^{-5} s
Number of subiterations	20

The SISS is noticeably more accurate than the CSS even for $n_{\mathrm{I}} = 1$, and the advantage becomes more obvious for $n_{\mathrm{I}} = 2$. It should be noticed that in all staggered schemes compared, the computation costs are consistent, and the parallelism advantage of the CPS only exists for the first sub-stage and thus negligible in this case (assuming a model runs faster than a solver).

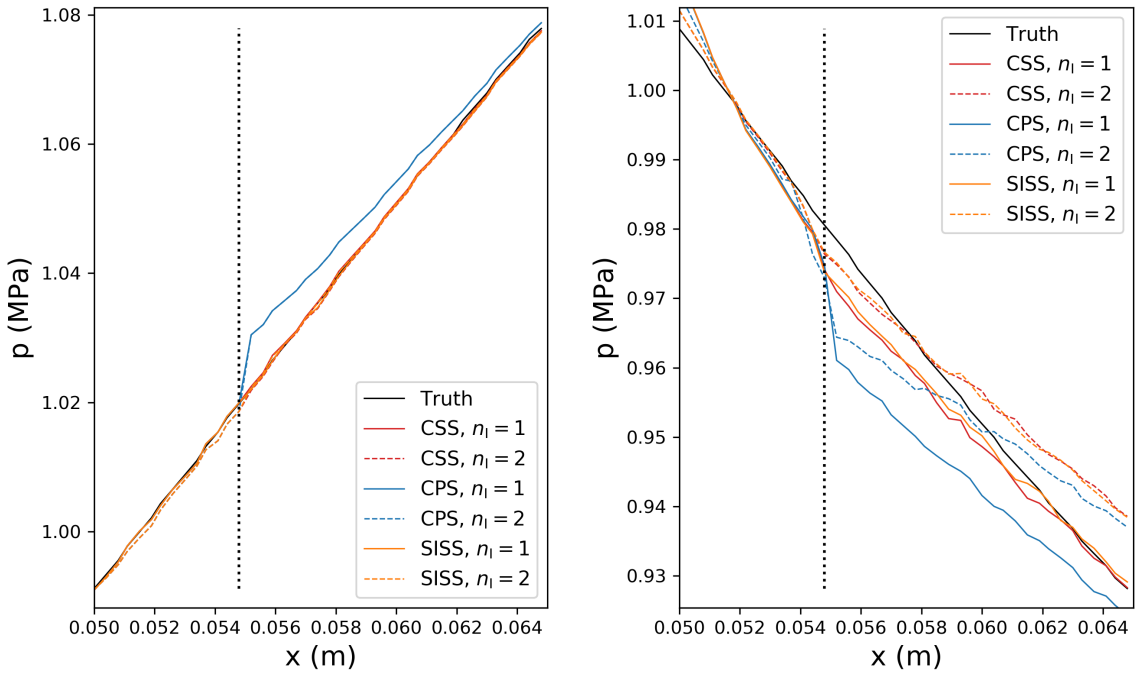


(a) $k_{\mathrm{S}} = 1$ (b) $k_{\mathrm{S}} = 25$

Figure 5.7: Zoomed-in results around the interface for *truth-solver* coupling with different staggered schemes. The vertical dashed line marks the interface.

5.4.2.2 Model-Solver Coupling

To further validate the choice in a more realistic setting, the experiment is repeated in a model-solver coupling setting. The ground truth feeder is replaced by a CPGNet identical to the one used in Sec. 4.6.1, except for the input and output sizes. The model is trained independently for $n_{\mathrm{I}} = 1$ and $n_{\mathrm{I}} = 2$ on a period covering 50 fine steps. The online prediction results are shown in Fig. 5.8. The behaviors of different schemes are qualitatively consistent with these in the previous truth-model coupling test. Moreover, a smaller error from the solver also in return leads to a smaller online modeling error across the interface in this test. The results illustrate the efficiency of the proposed SISS scheme.



(a) $k_{\mathrm{S}} = 1$ (b) $k_{\mathrm{S}} = 25$

Figure 5.8: Zoomed-in results around the interface for *model-solver* coupling with different staggered schemes. The vertical dashed line marks the interface.

5.5 Coupled Prediction for Viscous Burgers Equation

5.5.1 Problem Statement

In this section, we compare the intrusive ROM, non-intrusive ROM, and the CP-GNet in both a standalone and a coupled setting. The 2D viscous Burgers equation with periodic boundary conditions is studied. An important reason for using a periodic boundary condition is that even when no coupling is used, the evolution of the dynamics is self-contained, in contrast to the wave propagation problem used in Sec. 5.3.4, where the change in dynamics caused by the imposed boundary condition can only be inferred by autoregressive models with a long n_τ if not coupled with a solver. In this case, the use of coupling will not introduce benefits such as additional information input to the models, and the model behaviors in either the standalone setting or the coupled setting can be compared more directly, even for a short-n_τ model such as the MLP.

5.5.1.1 Domain Decomposition and Graph Generation

The domain decomposition for the coupled tests is illustrated in Fig. 5.9. The square-shaped domain is split along the vertical center-line into a left ROM part and a right FOM part. After the decomposition, two interfaces need to be coupled: the vertical center-line, and the left-right periodic boundaries. The top and bottom boundaries remain periodic and self-contained within a single type of model or solver, thus no coupling is needed.

For the graph used by the CP-GNet, the vertex (grid point in this case)-node mapping is used, with a special treatment for the periodic boundaries. The top-bottom periodic boundaries are used as the example and illustrated in Fig. 5.10. Below the bottom-most nodes, an additional layer of ghost nodes is added, whose values are kept updated to be the same as the up-most layer of nodes. Similarly,

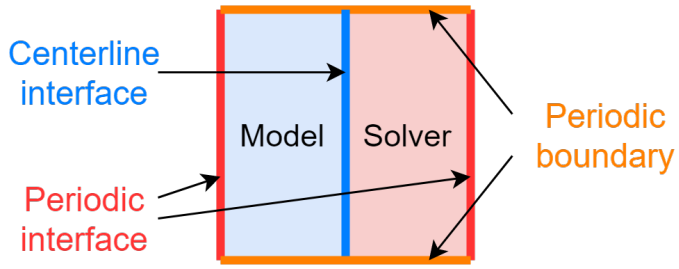


Figure 5.9: Schematic for domain decomposition. Two model-solver interfaces are present after the decomposition.

a ghost layer mirroring the bottom-most nodes is added above the top-most layer. This choice, instead of connecting the top-most and bottom-most nodes directly with edges, is to provide the edges with correct features, including the direction and length. The same is done for the left-right periodic boundary in the standalone setting.

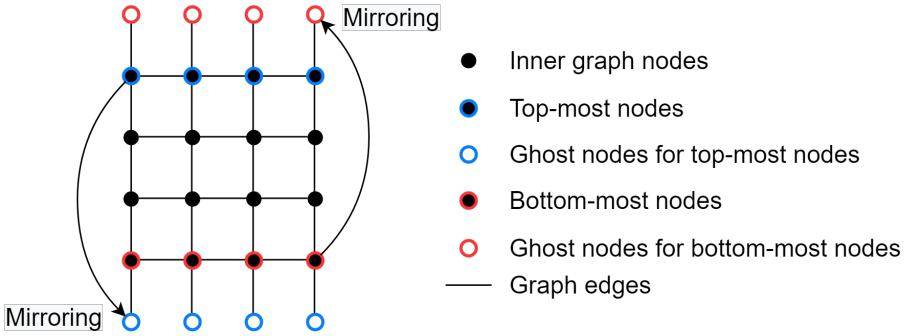


Figure 5.10: Schematic for ghost cells for the top-bottom periodic boundary.

5.5.2 Models

5.5.2.1 FOM

With periodic boundary conditions, the 2D viscous Burgers equation is given by:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + u\frac{\partial \mathbf{u}}{\partial x} + v\frac{\partial \mathbf{u}}{\partial y} - \nu\left(\frac{\partial^2 \mathbf{u}}{\partial x^2} + \frac{\partial^2 \mathbf{u}}{\partial y^2}\right) = 0, \\ x \in [0, W], y \in [0, H], t \in [0, T], \\ \mathbf{u}(0, y, t) = \mathbf{u}(W, y, t), \mathbf{u}(x, 0, t) = \mathbf{u}(x, H, t), \end{aligned} \tag{5.9}$$

where $\mathbf{u} = [u, v]^T$ is the 2D velocity vector, and ν is the diffusion coefficient. It must be pointed out that the set-up for this test problem is slightly modified from *Geneva and Zabaras* (2020), with parameters $W = 2, H = 2, \nu = 0.0025, T = 0.5$, and the IC is determined by a spatially periodic function $\mathbf{g}$:

$$\begin{aligned} \mathbf{g}(x,y) = \sum_{i=-L}^{L}\sum_{i=-L}^{L} \mathbf{a}_{ij}\sin(2\pi(ix+jy)) + \mathbf{b}_{ij}\cos(2\pi(ix+jy)), \\ \mathbf{u}(x,y,0) = \frac{2\mathbf{g}(x,y)}{\max_{\{x,y\}}|\mathbf{g}(x,y)|} + \mathbf{c}, \end{aligned} \tag{5.10}$$

where $\{\mathbf{a}_{ij}, \mathbf{b}_{ij}\} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_2), L = 4, \mathbf{c} \sim \mathcal{U}(-1, 1) \in \mathbb{R}^2$.

A finite-difference solver with a forward-Euler time integration scheme at $\Delta t = 0.001$ is used as the FOM. The spatial discretization is performed on a 64×64 grid point mesh using first-order upwinding for the convection, and central differencing for the diffusion. In the x-direction, the discretization is formulated as:

$$f^{\pm} \triangleq \frac{1}{2}\frac{u \pm |u|}{2}\mathbf{u}, \tag{5.11}$$

$$u\frac{\partial \mathbf{u}}{\partial x}\bigg|_{i,j} \triangleq \frac{f^{+}_{i,j} - f^{+}_{i-1,j}}{\Delta x} + \frac{f^{-}_{i+1,j} - f^{-}_{i,j}}{\Delta x}, \tag{5.12}$$

$$\nu\frac{\partial^2 \mathbf{u}}{\partial x^2}\bigg|_{i,j} \triangleq \nu\frac{\mathbf{u}_{i+1,j} - 2\mathbf{u}_{i,j} + \mathbf{u}_{i-1,j}}{\Delta x^2}. \tag{5.13}$$

The discretization in the y-direction can be easily obtained by substituting v for u, Δy for Δx, and index shifting in j for shifting in i.

5.5.2.2 RCMs

The POD-Galerkin intrusive ROM, the POD-MLP non-intrusive ROM, and the CP-GNet are compared. To demonstrate the capability of the non-intrusive model to its largest potential, automated hyper-parameter tuning is applied to the MLP. We consider a MLP model that can be described by the following 5 hyper-parameters:

1. Depth, an integer describing the number of hidden dense layers before the output layer.

2. Width, a shared integer describing the number of units per hidden layer.

3. USE_LN, a shared boolean describing whether to use layer normalization (*Ba et al.*, 2016) after a hidden layer.

4. USE_RES, a shared boolean describing whether to add residual connections between hidden layers.

5. Dropout ratio, the individual dropout (*Srivastava et al.*, 2014) ratio (can be zero) for each hidden layer.

Following the hidden dense layers, another dense layer with linear activation is used as the output layer, whose size is fixed to the number of POD modes n_r. Fig. 5.11 shows the architecture of the resulting MLP. The hyper-parameter is tuned with the HyperBand (*Li et al.*, 2017) algorithm based on the training convergence rate.

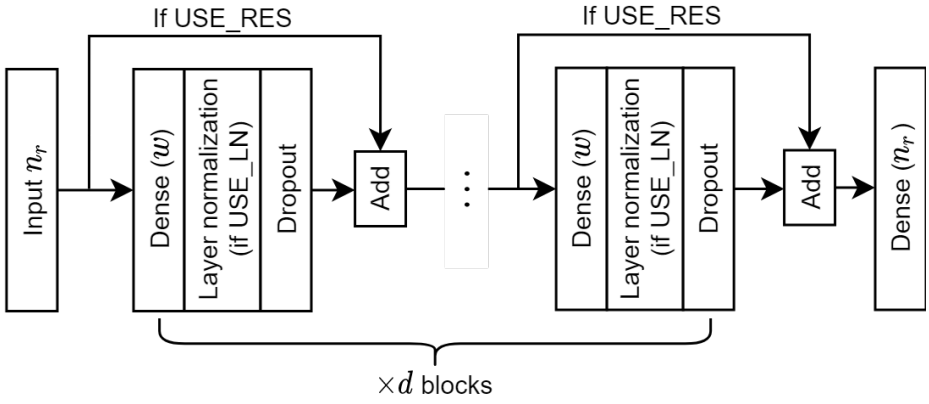


Figure 5.11: Architecture for MLP with tunable hyper-parameters.

For the CP-GNet, 5 processor blocks are used, each with 16-unit latent node features and 4-unit latent edge features.

5.5.2.3 Coupling

In this test, both sub-domains are time-integrated using the same forward-Euler scheme. For the coupling with an intrusive ROM, or a CP-GNet, the updated interface values are directly sent to the neighboring cells or corresponding graph nodes, as described in Sec. 5.2. For POD-MLP models, the interface flux projection method is used, which is shown to be the most accurate choice for this type of model in Sec. 5.3. For the present finite difference FOM with upwinding, the equivalent "flux" term in the x-direction, which is the one needed by the vertical interface, is given by:

$$\hat{\mathbf{F}}_{i+0.5,j} = (f^{+}_{i,j} + f^{-}_{i+1,j}) - \nu \frac{\mathbf{u}_{i+1,j} - \mathbf{u}_{i,j}}{\Delta x}. \tag{5.14}$$

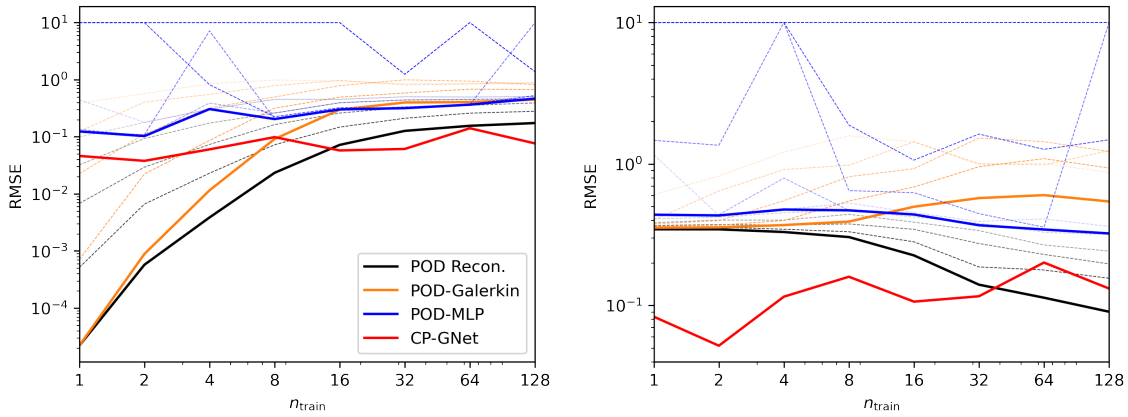
5.5.3 Results

129 FOM runs are generated, each with 501 snapshots including a random IC. Training is performed on the first 251 snapshots for 8 different numbers of training runs $n_{\text{train}} = \{1, 2, 4, 8, 16, 32, 64, 128\}$. For the POD-based ROMs, 5 different numbers of POD modes $n_r = \{5, 10, 20, 40, 80\}$ are studied. For each combination of n_{train} and n_r, the hyper-parameter tuning, as well as the model training, is conducted independently. The POD and models are also independent between the standalone setting and the coupled setting. CP-GNets are also independently trained for each n_{train}.

For each model, testing is conducted by performing a 500-step rollout prediction from the IC for runs #1 (always within training) and #129 (always outside training). It should be noted that even for run #1, the second half of the prediction is beyond the training period. The RMSE for the rollout predictions are plotted in Fig. 5.12, where the errors for the first and second halves of run #1 are reported separately, and a single full-period error is reported for run #129. Some models are numerically unstable in the online prediction, and the RMSE is clipped to 10 in these cases. For the POD

models, combined curves for the lowest possible RMSE at each n_{train}, as well as the individual curves for each n_r are plotted. It is obvious that for the POD reconstruction and POD-Galerkin ROM, the error decays with n_r almost monotonically under both seen and unseen conditions. However, for the POD-MLP, many combinations of n_r and n_{train} are unstable, although at least one stable model has been obtained for each n_{train}. This reflects a common problem with many deep learning models for long rollout predictions - additional treatments such as regularization and noise injection are necessary to improve the robustness, which are often developed heuristically. If we focus on the best possible performances, the non-intrusive ROM shows a close level of error to the intrusive one. In contrast, the CP-GNet remains stable in all cases beyond $n_{\text{train}} = 4$, which is a rather small amount of training data. Actually, the CP-GNet provides a lower testing RMSE than any POD model on the unseen data once it is sufficiently trained. In most cases, the online CP-GNet rollout prediction is even more accurate than the offline POD projection-reconstruction.

Fig. 5.13 shows contours for u as qualitative results for the rollout predictions. For each type of model, the combination of hyper-parameters with the lowest RMSE for run #129 is selected. Either POD model is only able to predict a visually similar-to-truth field within the training period, whereas the GP-GNet result is close to the truth in all conditions. The gap between different models is especially significant in the coupled results for run #129, where a clear discontinuity in the centerline interface is visible for the POD-based models.



(a) Standalone, run #1, first half (seen)

(b) Standalone, run #1, second half (unseen)

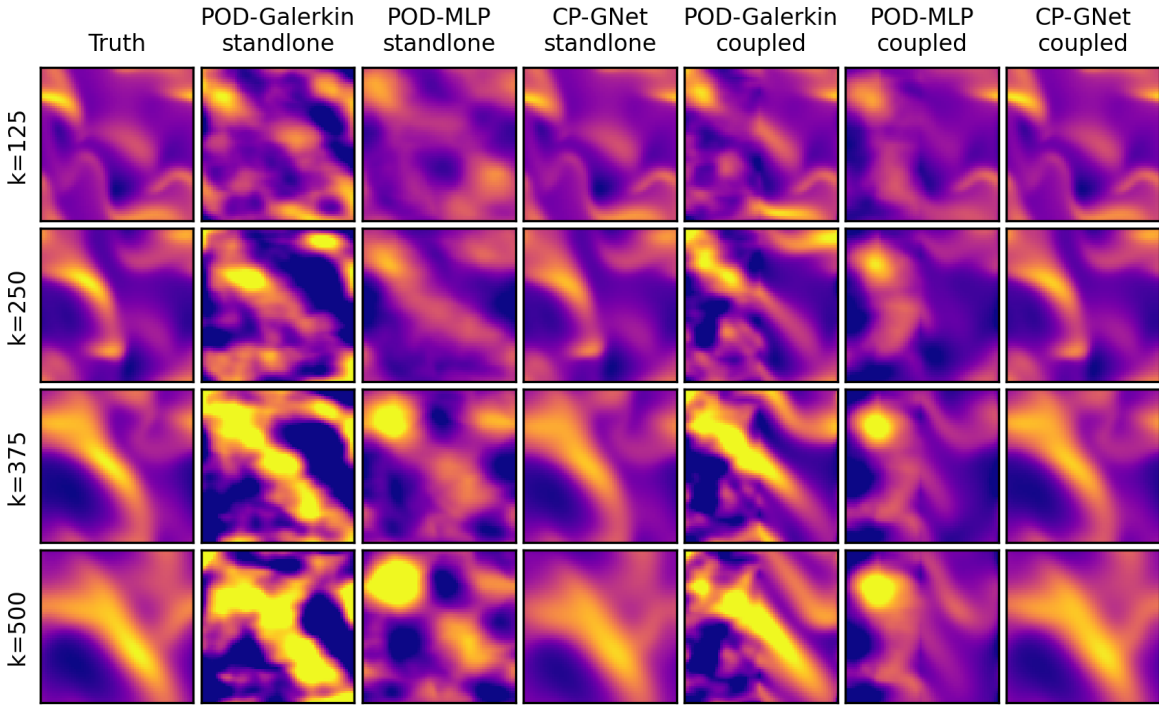
(c) Standalone, run #129, full period (unseen)

(d) Coupled, run #1, first half (seen)

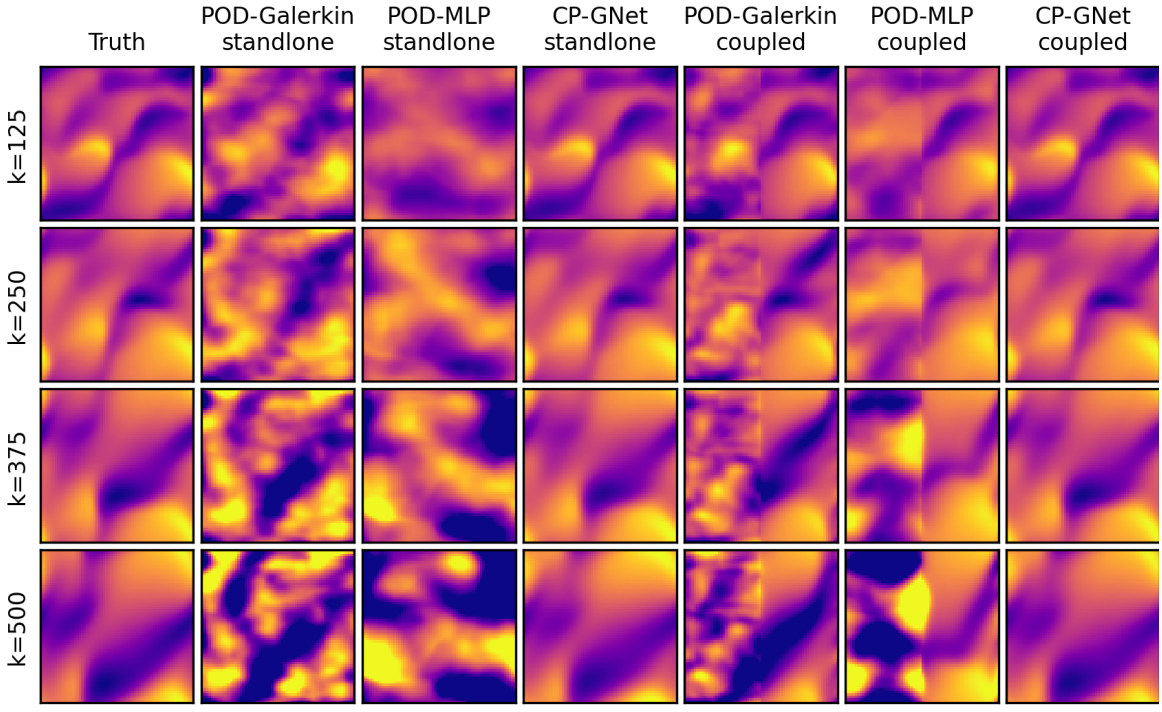
(e) Coupled, run #1, second half (unseen)

(f) Coupled, run #129, full period (unseen)

Figure 5.12: RMSE for online rollout predictions and POD reconstructions (Recon.), with numbers above 10 clipped. For the POD-based results, the solid line marks the minimum error across all n_r at a given n_{train}. The dashed lines with different opacity show the results for individual n_r, with a higher opacity denoting a larger n_r.



(a) Run #1



(b) Run #129

Figure 5.13: Contours for u in rollout predictions. For each type of model, the best-performing hyper-parameter is shown.

5.6 Coupled Prediction for a Single-Injector Rocket Engine Combustor

5.6.1 Problem Statement

As the final test case of the thesis, the framework is applied again to a single-injector rocket engine combustor, which echos back to the initial demonstration of the framework in Chapter II. The 2D numerical test is adopted from *Huang* (2022), which can be viewed as a high-fidelity and complete combination of the quasi-1D model used in Chapter II, and the reacting injector used in Sec. 4.6.1. As in our quasi-1D study, the single-injector rocket engine combustor is decomposed into two sub-domains, with the first, reaction-intensive sub-domain to be predicted by a RCM, and the second, geometry-variable sub-domain to be predicted by a solver. The most promising combination of RCM and coupling scheme in the thesis, CP-GNet and SISS are applied.

5.6.2 Domain Decomposition and Reduced-Domain Training

The FOM-related settings for this test, including the domain decomposition and reduced-domain training, are adopted from *Huang* (2022). As shown in Fig. 5.14, 3 geometries are studied in total, including a truncated one for the generation of training data, and two complete ones of different lengths for testing. The geometry for the first sub-domain, separated by an interface at $x = 0.0551$ m, is shared by all three cases. In the training case, a straight channel is appended after the interface as a buffer region, in which the grids coarsens gradually in the x-direction. The end of the channel is taken as a characteristic boundary, on which span-wise uniform characteristic perturbations are applied. The effectively-1D perturbations gain 2D features through the buffer region, and representative unsteady behaviors are triggered in the first sub-domain. In the two testing geometries, downstream chambers of different

lengths L_c are appended, followed by a shared nozzle. As discussed in Chapter II, different acoustic profiles, and consequently different coupled flow dynamics. More detailed parameters are provided in Table. 5.2.

As listed in the same table, two sets ($n_f = 1$ and $n_f = 2$) of characteristic perturbations are applied in two independent training runs. In the first set, a 10% single-frequency perturbation at 1150 Hz is applied. As estimated in *Huang* (2022), this approximately corresponds to the resonant frequency for the longer testing geometry. In practice, it is hard to include a corresponding perturbation frequency for each geometry to be predicted. To evaluate the performance under this restriction, a second training set with two offset frequencies, 1100 Hz and 1300 Hz, is used. The magnitude for each frequency is reduced to 5% to maintain the super-imposed maximum amplitude consistent with the single-frequency case. The estimated resonant frequency for the shorter testing geometry is 2200 Hz, which deviates significantly from the training frequencies, and can be regarded as an off-design case.

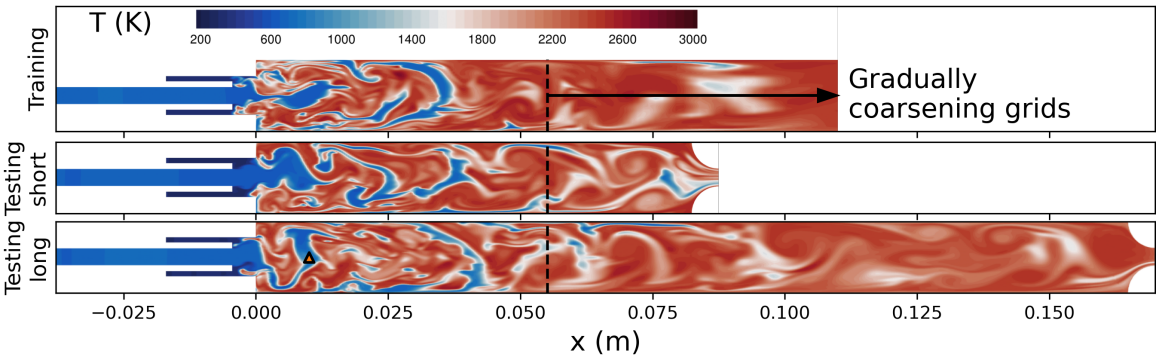


Figure 5.14: Experimental geometries. Contours are for temperature. The orange triangle marks the probe location.

5.6.3 Model Details

The FOM solutions are conducted with the GEMS solver, with a laminar flamelet combustion model (*Peters*, 1988). The model relies on a pre-generated look-up table, and represents the detailed combustion processes with locally 1D flamelet structures.

Table 5.2: Geometry parameters.

Case	Training	Testing, short	Testing, long
n_x in first sub-domain	37340		
n_x in second sub-domain	9400	19740	47940
Chamber length (m) (backstep to the end of geometry)	0.11	0.0876	0.17
Perturbation frequency (Hz)	$n_f = 1$: 1150 $n_f = 2$: 1100 + 1300	N/A	N/A
Estimated resonant frequency (Hz)	N/A	1150	2200
Span	0.2 ms/200 steps	0.8 ms/800 steps	

In each 1D structure, properties such as chemical species fractions and temperature are assumed to be completely described by two parameters corresponding to the dimensions of the look-up table: 1) the mixture fraction describing the fuel-to-oxidizer ratio [3], and 2) the progress parameter describing the non-equilibrium state. In addition to the 2 flamelet variables, 4 other variables, p, u, v, T are also solved. A time-step size of $\Delta t = 1 \times 10^{-6}$ s is used, with 20 DTS sub-iterations included in each step. For each training set, only 200 snapshots are included, which spans approximately 25% of an oscillation cycle for the longer testing geometry. 800 snapshots are generated as the ground truth for each testing case.

A 10-processor-block CP-GNet architecture with 56-unit latent node features is used. Except for changes in the latent dimension and input/output sizes due to the switch to a flamelet model, the architecture is similar to the one in Sec. 4.6.1. The model is trained to work at a time-step size twice coarser than the one for FOM. Two identical models CP-GNet$_1$ and CP-GNet$_2$ are trained independently for the single- and dual-frequency training sets, respectively, which are distinguished by subscripts "1" and "2" in the rest of the section.

In addition to model-solver coupling, a few other configurations are also studied as listed in Table 5.3. First, in order to evaluate the predictive capability of the CP-GNet for the reduced-domain independently, the model is coupled with a "ground truth

[3]The mixture fraction can be further separated into a mean and a variance term. In the current study, a zero-variance model is used, and thus a single variable is sufficient.

feeder", similar to that used in Sec. 5.4.2.1. The feeder loads in the pre-computed testing data for the downstream sub-domain at an interval equivalent to the time-step size of the upstream CP-GNet, i.e. $\Delta t = 2 \times 10^{-6}$ s, and send the interface values to the model in the online prediction. This test narrows the source of error to the CP-GNet only. Secondly, the CP-GNet is shown to be portable and predictive for unseen geometries in Sec. 4.6.2, which leads to our exploration on the possibility that the model trained for the reduced-domain can be directly used for full-domain predictions without coupling with a solver. What adds value to the exploration is that the geometric and flow-dynamic features in the downstream nozzle are largely different from these covered in the reduced-domain training, which leads to a more challenging task compared with predictions between moderately varied geometries, such as the one in Sec. 4.6.2 and similar ones in related work. Lastly, full-domain POD projection-reconstructions are performed for each testing case. For each geometry, an individual set of 100-mode POD bases is obtained from the first 200 snapshots of the pre-computed solution. The reconstruction results are supposed to reflect the upper limit of any classic full-domain POD ROMs.

Table 5.3: Test configurations.

Test configuration	Model/solver	
	First sub-domain	Second sub-domain
Truth	GEMS, $\Delta t = 1 \times 10^{-6}$ s, P=20	
CP-GNet+Truth	CP-GNet$_1$, $\Delta t = 2 \times 10^{-6}$ s	Ground truth feeder, $\Delta t = 2 \times 10^{-6}$ s
CP-GNet+GEMS	CP-GNet$_1$/CP-GNet$_2$, $\Delta t = 2 \times 10^{-6}$ s	GEMS, $\Delta t = 1 \times 10^{-6}$ s, P=20
CP-GNet (full-domain)	CP-GNet$_1$, $\Delta t = 2 \times 10^{-6}$ s	
POD	Offline POD projection-reconstruction	

5.6.4 Results

The results of the prediction are interpreted based on two representative variables, pressure p and temperature T. More specifically, profiles for the average pressure in

the x-direction are plotted in Fig. 5.15 and 5.18 for the longer and shorter testing geometries, respectively. Contours for the temperature field are present in Fig. 5.16 and 5.19. In each of the figures, 4 snapshots based on the finer (solver) time-step, $k = \{100, 200, 400, 800\}$, are present. For a more continuous visualization, a probe monitor is placed 0.01 m downstream the span-wise center of the back-step, which is marked in Fig. 5.14. The probed histories for p and T are plotted in Fig. 5.17 and 5.20. In Appendix B.2, contours for the rest variables are present, which further supports the observations below.

CP-GNet$_1$ + truth. Starting from the "in-design" case that couples CP-GNet$_1$ with the truth feeder for the longer testing geometry. In this case, the 1150 Hz characteristic perturbation in the reduced-domain training ideally aligns with the acoustic response frequency of the chamber. In this single-directional coupling (from truth to model) test, the model's capabilities of processing unseen downstream interface conditions, and predicting the future states in the designed reduced-domain are evaluated. The former is validated mainly by the centerline pressure profiles, where the CP-GNet follows the downstream truth closely, with a small gap at the interface that saturates after 200 model prediction steps (shown as $k = 400$ based on solver steps). For the latter, a strong future-state predictive capability is demonstrated by the temperature contours and the probed histories. It can be observed that the prediction results match the truth closely up to $k = 400$, which is already twice longer than the training period. At $k = 800$, the deviation from the truth becomes more visible, and local extreme values start to occur.

CP-GNets + GEMS. When the coupled truth feeder is replaced with GEMS in online prediction, a bi-directional coupling system is formed. The prediction is this situation is more challenging as the downstream responses would deviate even more from the training. Encouragingly, CP-GNet$_1$ is still able to generate similar results

in the upstream sub-domain. Meanwhile, with online updates in the downstream sub-domain through SISS, the pressure profiles connect smoothly across the interface. Inevitably, the downstream flow is not as accurate as predicted by a complete full-domain FOM, demonstrated by blurred small-scale structures. Nevertheless, the large-scale features are predicted reasonably well, which is encouraging for the future applications of the framework, as the coupled FOM is supposed to be low-cost and low-fidelity by design. With 2 offset frequencies used in the training, the performance of CP-GNet$_2$ deteriorates slightly compared with the CP-GNet$_1$, which is reflected by an earlier onset of visible errors at $k = 200$, and larger deviations in the probed histories.

Full-domain CP-GNet$_1$. When CP-GNet$_1$ trained in the reduced-domain is directly applied for a full-domain prediction, a similar result in the upstream region as in the coupled predictions is observed up to $k = 400$. The prediction for the majority of the downstream chamber is actually even more accurate than a coupled GEMS solver at $k = 100$, which clearly illustrates the generalizability advantage of a local CP-model to unseen geometries and a reasonably wide range of dynamic patterns. However, the model is unable to make accurate predictions in the nozzle region, in which the flow dynamics are tremendously different from these covered in the training data. The error in the nozzle region gradually propagates upstream, and ultimately "pollutes" the full domain towards the end of the testing period.

Off-design test case. In the shorter testing geometry, for which the resonant frequency is not included in either training case, the predictive capability of the CP-GNet is challenged. The model is still able to predict accurately up to $k = 100$ in all coupling configurations. However, the gap in the pressure profiles across the interface grows and saturates much earlier when coupled with the truth feeder. When coupled with a solver, the bi-directional coupling with SISS is able to keep the pressure profiles

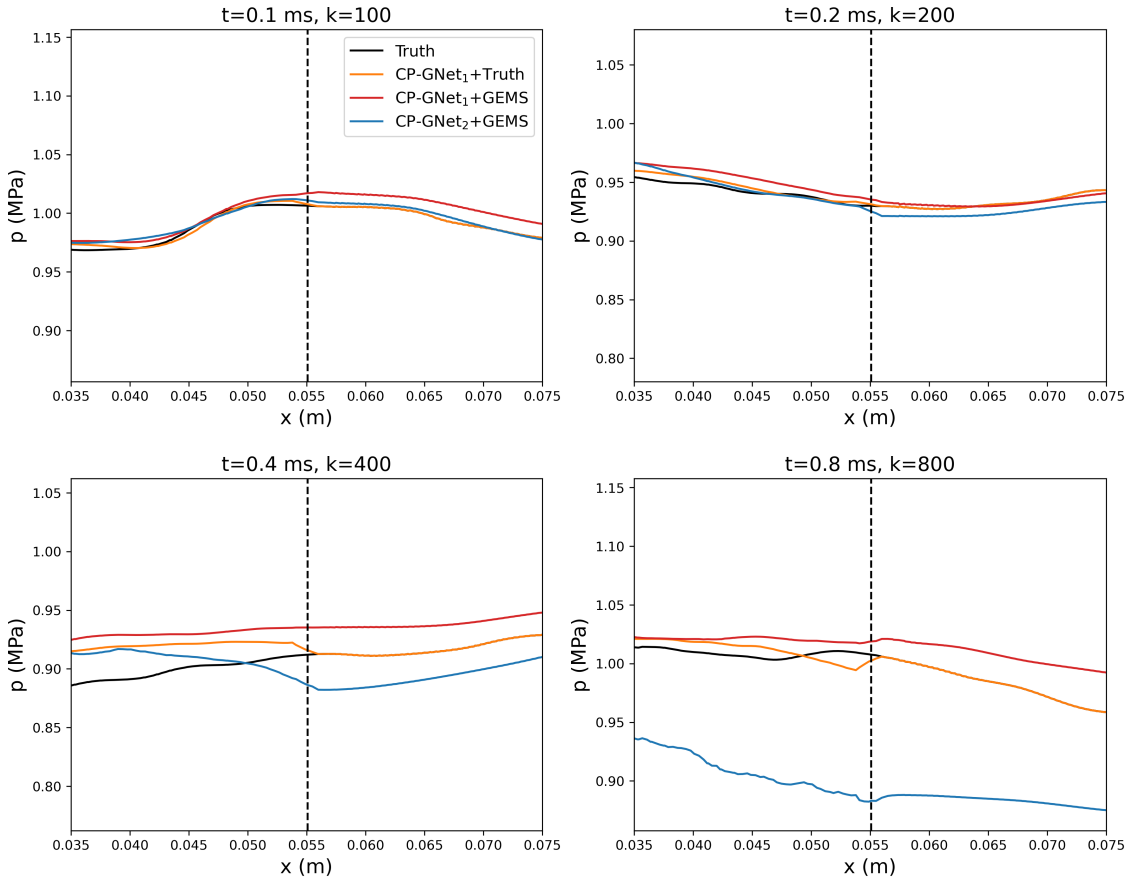


Figure 5.15: Average pressure on the x-direction for the longer (in-design) testing geometry. Zoomed-in around the domain interface, marked by the vertical line.

as well as flow structures consistent across the interface. As both training sets are away from the target resonant frequency, the difference between their results is less clear, with visible errors starting to present at $k = 200$ in the temperature fields for both CP-GNet$_1$ and CP-GNet$_2$. The off-design results show that even for a highly portable and training-data-efficient model such as the CP-GNet, the representativeness of the training data still has a strong impact on the predictive capabilities, and a careful design of the training FOM is necessary.

POD projection-reconstruction. In both test cases, POD is able to perform a perfect reconstruction of the training period up to $k = 200$, yet becomes ineffective quickly afterward. Indeed, it is hard to observe any meaningful features in the POD

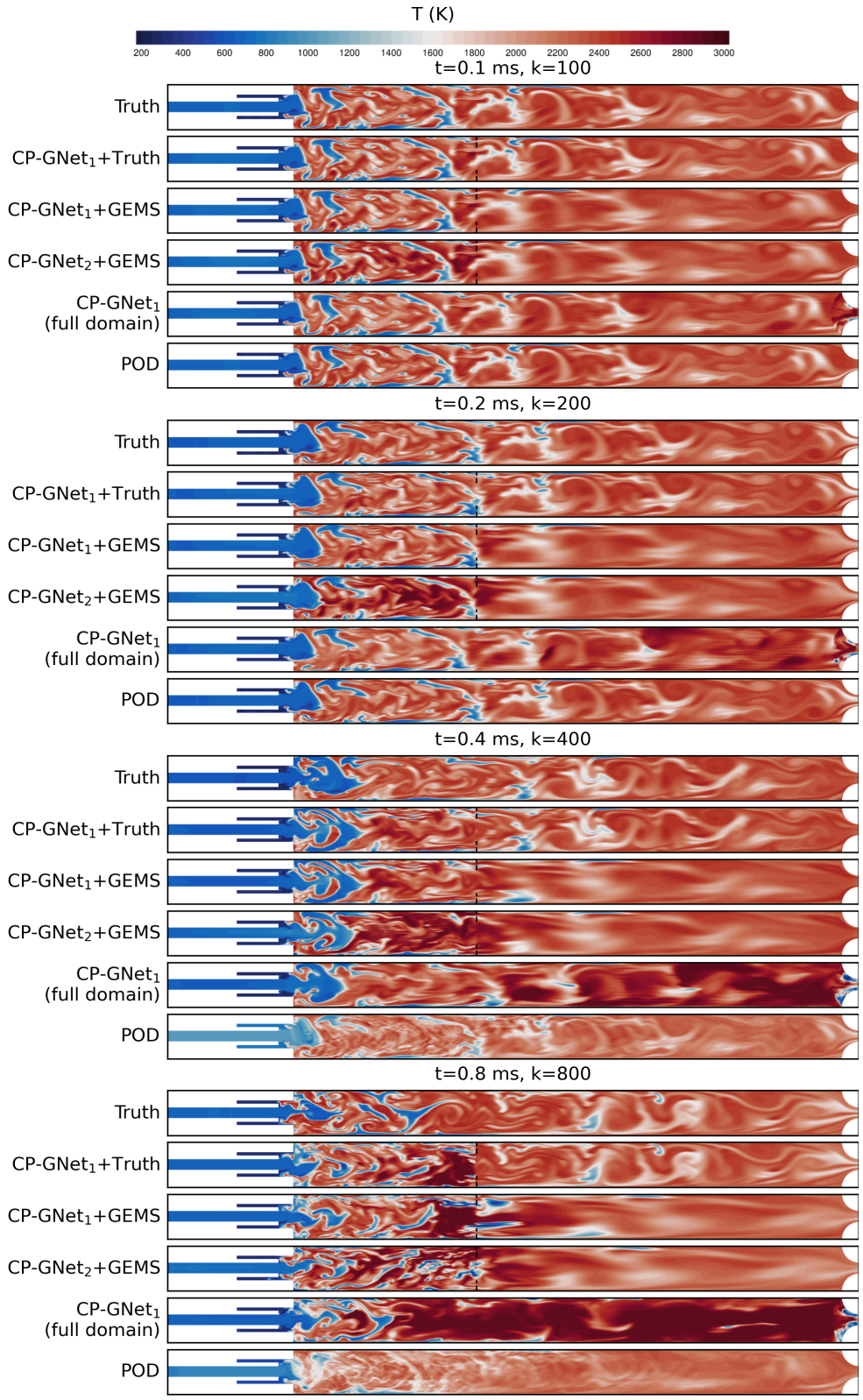
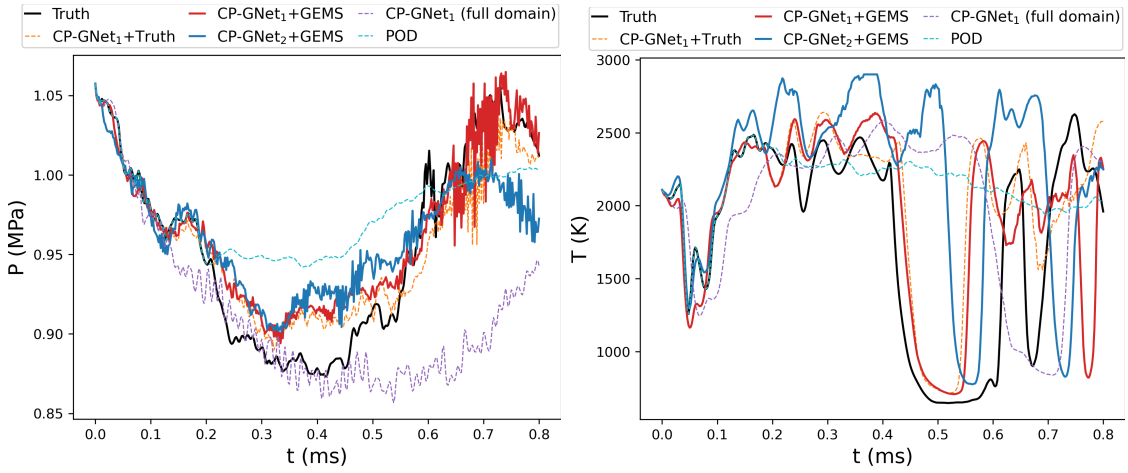


Figure 5.16: Temperature contours for the longer (in-design) testing geometry.



(a) Pressure (b) Temperature

Figure 5.17: Probed histories for the longer (in-design) testing geometry.

reconstruction results at $k = 400$ and $k = 800$. The probed histories also show an immediate deviation after the end of the training. The sharp contrast between the training and testing stages serves as a clear example of overfitting for global dimensionality reduction in a low data condition, and reflects the training-data-efficiency advantage of a local model.

Timing. For a direct comparison, timing results for the full-domain prediction of the shorter testing geometry are used. A CP-GNet covers the 0.8 ms testing period with 400 time-steps, which takes 131 seconds on one Nvidia RTX A6000 GPU. In comparison, a FOM simulation with GEMS for the same period involves 800 time-steps, which takes over 6490 seconds on 60 CPU cores.

5.7 Summary

This chapter mainly focused on explorations of interface models applicable to non-intrusive models, and staggered procedures for coupling mismatched time-integration schemes. In the former, three types of models, flux projection, interface MLP, and overlapped projection, are proposed and compared. The flux projection model deals

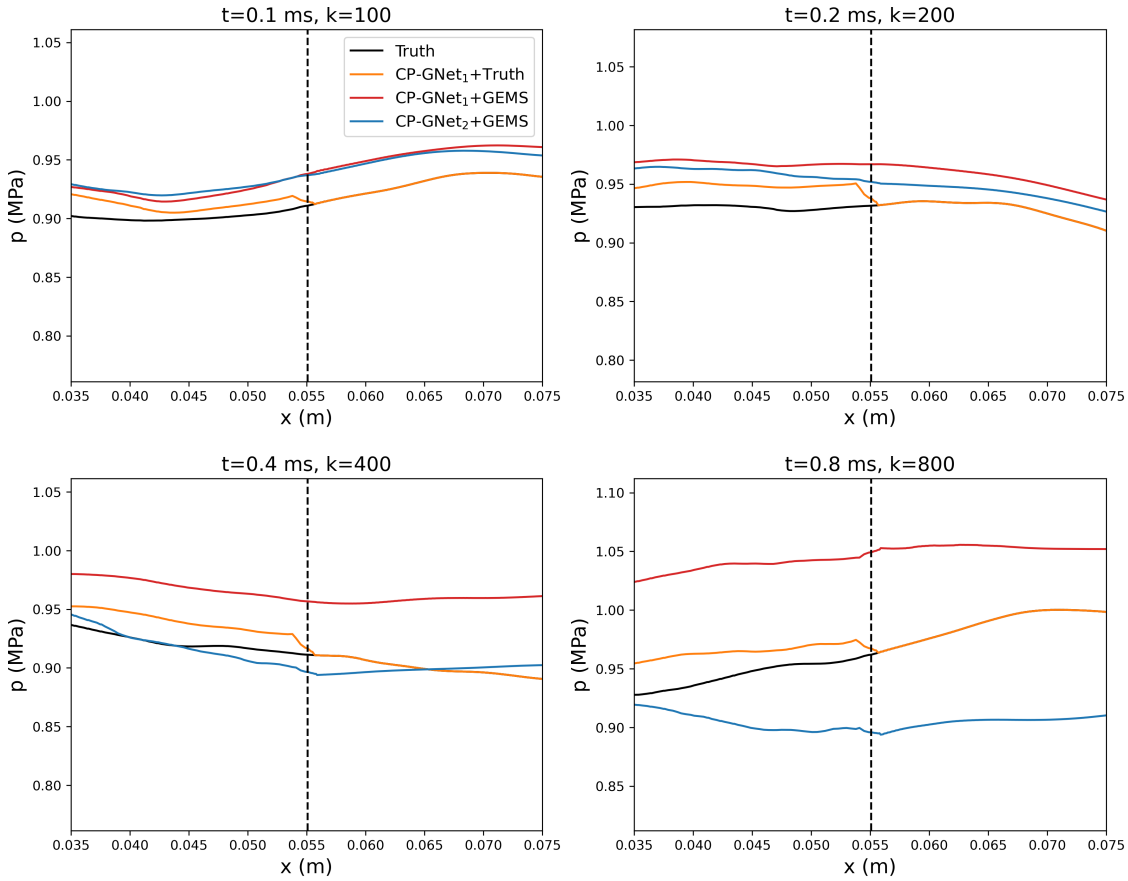


Figure 5.18: Average pressure on the x-direction for the shorter (off-design) testing geometry. Zoomed-in around the domain interface, marked by the vertical line.

with the interface communication in an intrusive manner by computing the formal flux term, and which is then projected onto the basis for the interface cell in order to update the reduced order variables. The interface MLP bypasses the need for governing equations, and models the flux term implicitly with a MLP that takes variables in the internal and external interface cells as the input. The overlapped projection method extends the projection bases into the solver sub-domain, such that the coupling is directly enforced in the projection process. The interface models are applied to two non-intrusive models: MLP and LSTM, and tested on the same wave propagation problem from Sec. 3.4, on which the non-intrusive models have been studied in a standalone manner. The flux projection and interface MLP are shown to effectively communicate with the non-intrusive models, enabling the MLP to match

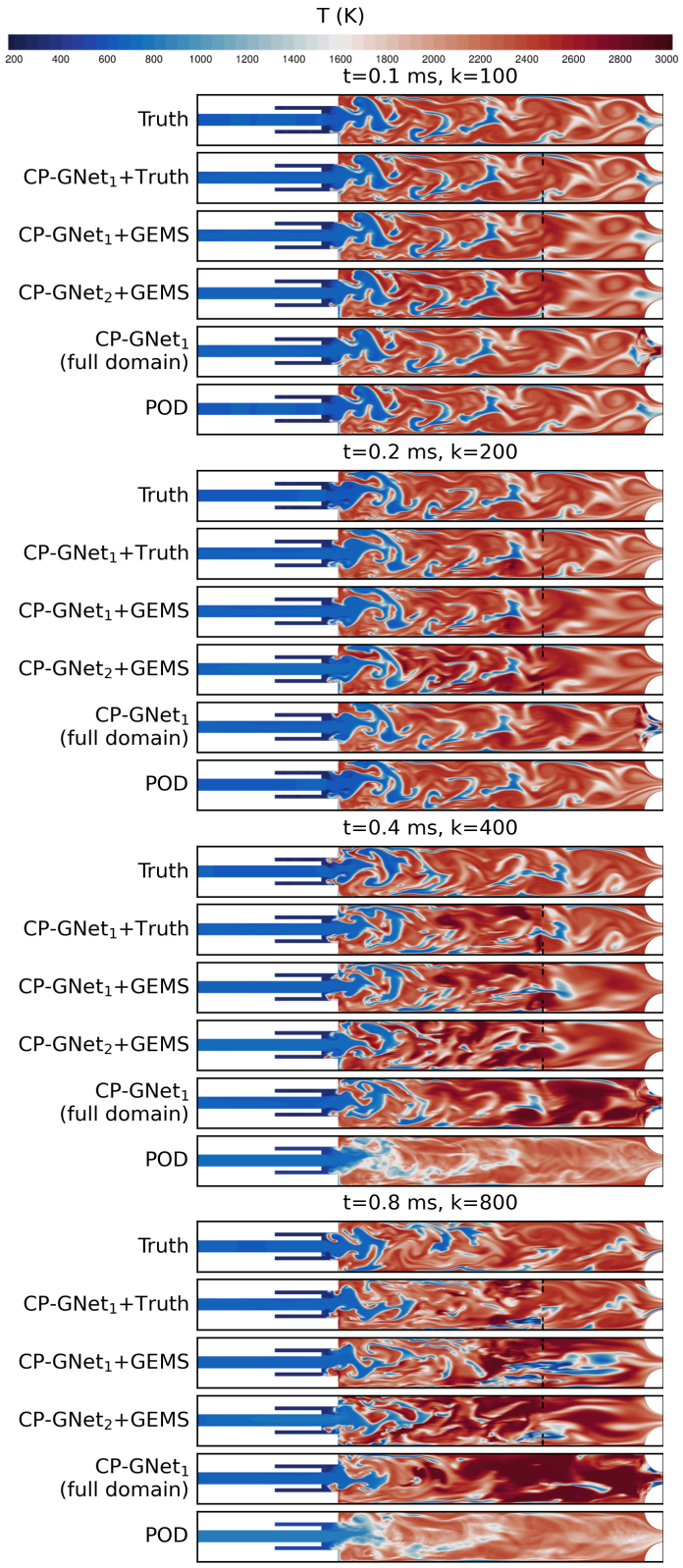
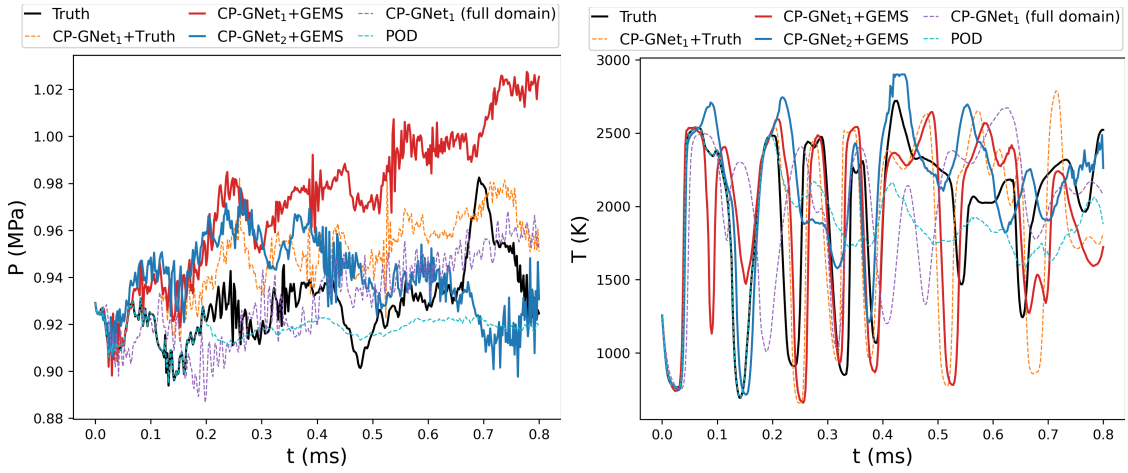


Figure 5.19: Temperature contours for the shorter (off-design) testing geometry.



(a) Pressure (b) Temperature

Figure 5.20: Probed histories for the shorter (off-design) testing geometry.

with the FOM solver in the prediction of the constantly varying wave frequency, which was impossible when no coupling was applied in Sec. 3.4. Through a joint training, the interface MLP is also able to further improve the performance of the LSTM model.

In the study of staggered schemes, two popular procedures in fluid-structure interactions, CSS and CPS, and a specialized improvement for model-solver coupling, SISS, are introduced. Among the three, CPS is the most straightforward one as the communications in both directions take place simultaneously at the beginning of each solver time-step. However, it is also shown to give the largest error in our coupling test between a model that takes a coarser time-step without any sub-iteration, and a solver with a large number of sub-iterations ($P = 20$). Both the CSS and the SISS advances the model before the solver, and are shown to significantly reduce the interface error compare with the CPS. By extending interpolations and communications to the sub-stage level, the SISS shows a even better performance than the CSS.

With the findings in this and preceding chapters, the most promising RCMs, interface model, and coupling scheme are integrated and demonstrated in two coupled tests. In the test with 2D viscous Burgers equation, an intrusive POD-Galerkin

ROM, a non-intrusive POD-MLP ROM, and a CP-GNet model are compared in both standalone and coupled settings. With the help of hyper-parameter tuning, the POD-MLP is able to outperform the POD-Galerkin in multiple cases. The flux-projection interface method is also shown to enable the non-intrusive model to couple effectively with the FOM across both interfaces, resulting in a guided flow pattern. However, the POD-MLP occasionally suffers from stability issues across all training configurations, despite the hyper-parameter tuning applied. In contrast, the CP-GNet remains stable and provides the lowest prediction error once sufficiently trained (beyond $n_{\text{train}} = 4$). Indeed, the online CP-GNet rollout prediction is even more accurate than the offline POD projection-reconstruction in most cases. In the final test case of a 2D CVRC, SISS is used to couple a CP-GNet with a high-fidelity FOM solver across different time-step sizes and numbers of sub-stages. With a single training process based on limited provided data from a FOM for the reduced-domain (plus a buffer region) with a characteristic boundary and perturbation, the CP-GNet is shown to be predictive not only for the training sub-domain, but also for a large downstream region before the nozzle. With the help of the coupled solver, the integrated framework is shown to be predictive at different chamber lengths for a much longer period than the training period.

CHAPTER VI

Conclusions and Perspectives

This thesis is composed in the background of a "low data regime" that lies at the heart of the contradiction between the challenges in training a data-driven Reduced-Complexity Model (RCM), and its supposed benefits. We use the term RCM to broadly define reduced order models as well as surrogate techniques that can model spatio-temporal fields of variables. The low data regime is a consequence of the fact that the training data available to the RCM may be limited in spatial and temporal coverage and parametrically sparse. In fact, considering the motivating example of a rocket combustor, existing high-fidelity methods are unaffordable even to perform a single full-system simulation. Against this backdrop, a RCM demands the training data to cover a large-enough envelope in the hyper-parameter space to be predictive in many-query scenarios.

In the present work, a number of ideas were proposed towards the end of addressing the development of RCMs in the low data limit. In Sec. 1.4.1, the framework of a full-system modeling strategy in this regime was described. The framework is multi-fidelity, which uses a data-driven RCM and its mirrored copies for the regions with complex physics that require a high-fidelity solution, in coupling with a solver for the rest of the domain with simpler physics that can be solved with a lower fidelity, but with variable geometries that cannot be easily handled by a RCM. Fig. 1.3 (replicated

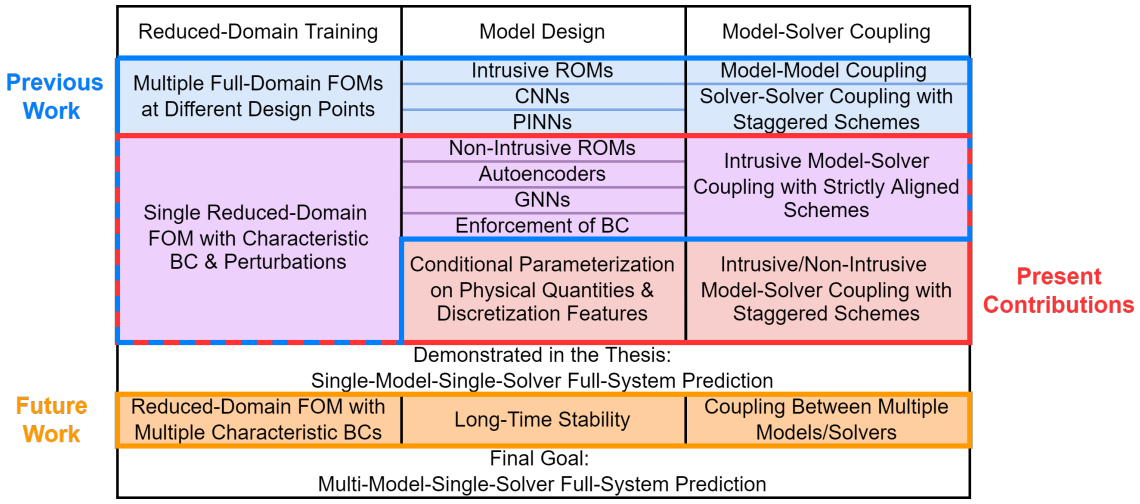


Figure 1.3: Contributions of the thesis in the roadmap towards full-system modeling (replicated).

below) shows the scope of this work in the roadmap towards the realization of the framework. The techniques involved in the roadmap cover previous, current, and future work, and can be categorized into three main aspects:

1. Reduced-domain training: training data generation in an isolated reduced-domain.

2. Model design: design of efficient and predictive RCMs

3. Model-solver coupling: effective interfacing methods to couple a data-driven RCM and a high-fidelity solver.

6.1 Reduced-Domain Training

In this aspect, the characteristic training method first introduced by *Huang et al.* (2016) is used in the study of a quasi-1D model for the Continuously Variable Resonance Combustor (CVRC) in Chapter II. The Proper Orthogonal Decomposition (POD) bases generated from a single training set with broadband characteristic perturbations is shown to provide a comparable or even lower projection-reconstruction

error than a specifically trained one using the traditional full-domain training method at different testing geometries. The advantage is further verified in the online prediction tests and off-design evaluations with an intrusive POD-Galerkin ROM. In the final test case of a 2D version of the CVRC in Sec. 5.6, data generated from a similar manner for the upstream sub-domain is used to train a Conditionally Parameterized Graph Neural Network (CP-GNet), which is then shown to be predictive not only for the training sub-domain, but also for a large downstream region before the nozzle. The same test is repeated with offset perturbation frequencies, as well as an off-design testing chamber length with a significantly different resonant frequency. In these additional tests, the model performance noticeably deteriorates, which emphasizes the importance of a carefully designed training configuration, even for a highly portable model.

6.2 Model Design

The RCMs studied in this thesis fall into two broad categories, the non-intrusive ROMs in Chapter III, and the local surrogate models in Chapter IV, distinguished by whether a global dimensionality reduction is performed. The reduction serves as the first step in ROM procedures, followed by a second step in which the resulting reduced order variables are computed as an approximation to the high-order FOM. Bypassing the formal intrusive computation using projected governing equations, non-intrusive ROMs benefit from a higher flexibility in the choices for both steps. By replacing the POD with a convolutional autoencoder in the first step, improvements in both reconstruction fidelity and final prediction accuracy are demonstrated in both test cases (future-state prediction for wave propagation, parametric prediction for flow over a cylinder). Independently of the first step, the second step can be replaced by either an autoregressive model for future-state predictions, or a combination of a temporal dimensionality reduction model and a regression model for parametric

predictions of a fixed spatio-temporal domain. The efficiency gain in the second type of model is especially significant as it bypasses not only the expensive computation of the high-order nonlinear term per time-step, but also the time-integration procedure. The advantage is clearly demonstrated in the flow-over-cylinder case, in which the multi-level autoencoder networks achieved a more than 2500-time acceleration, with an error below 2%.

Linearly or nonlinearly projected, intrusively, or non-intrusively computed, a limitation exists for most ROMs, that a global dimensionality reduction model, as well as the ROM based on it, is fixed to the training geometry and range of dynamics, which is demonstrated in Sec. 4.3 on an advection-diffusion problem. Although our framework can effectively alleviate the restriction on the geometry using domain decomposition and coupling, the training data still needs to be sufficient to cover the possible range of dynamics. In the low data regime that this thesis is set for, this may be a serious drawback as shown by the large POD projection errors for the 2D CVRC in Sec. 5.6.

A more flexible and less training-data hungry type of model is the local surrogate model, most popularly exemplified by Convolutional Neural Networks (CNNs). A local surrogate model performs local computations through individual discrete points with the same set of parameters in a "swiping-through" manner, without generating a fixed mapping to the global coordinates. Due to the weight-sharing property, each spatial point in a snapshot becomes an effective training sample for the model, which leads to a tremendous reduction in the total number of training snapshots needed. To deal with irregular, non-Euclidean features, Graph Neural Networks (GNNs) are widely used in the place of CNNs. As described in Sec. 4.4.1, existing spatial discretization methods in CFD can be easily mapped to graphs suitable for GNNs. However, GNNs often concatenate graph-node features with graph-edge ones, ignoring their heterogeneous natures. The naive treatment, along with similar approaches

such as a brute-force fitting of high-order terms, undermines the credibility of current GNN architectures as a simulation tool, which has much higher accuracy demands than other tasks of GNNs such as classification ones. This has motivated our explorations on conditional parametrization, and ultimately led to the development of the CP-GNet, which can efficiently learn the hierarchical and/or high-order relations between heterogeneous features by making model parameters for certain features into trainable functions of other ones. The CP-GNet, and other CP-modified models, are shown to bring significant improvements to their non-CP versions and other state-of-the-art baselines in multiple tasks including closure modeling (Appendix A.1), super-resolution of turbulent flow (Appendix A.2), and flow simulations with chemical reactions (Sec. 4.6.1, Sec. 5.6), as well as on unseen irregular meshes (Sec. 4.6.2, Sec. 5.6). The improvements are especially worth noting given that the form of CP modification to existing layers, described in Sec. 4.2, is relatively straightforward.

6.3 Model-Solver Coupling

It is mentioned in Sec. 1.4.2 that the specialized coupling method between RCMs and solvers has not been actively explored. Indeed, the necessity of such methods is directly reflected by the obvious improvement brought by the interface models to the non-intrusive models used in the wave propagation problem. In Sec. 3.4, neither the frequency nor the amplitude can be predicted by a simple POD-MLP model. In Sec. 5.3.4, visible errors are eliminated when a proper coupling method is added to the same model. We also demonstrated the benefit of a joint training of an "inner" model and an interface model, which can produce an even lower error than the formal flux projection procedure when applied to complex autoregressive models.

Regardless of the complexity in the interface model itself, additional treatments are necessary when time-step sizes or numbers of sub-stages are different between the time-integration schemes across the interface. In this work, three staggered pro-

cedures are compared, including two existing ones, Conventional Serial Staggered (CSS) and Conventional Parallel Staggered (CPS), and a specialized modification, Sub-Iteration Serial Staggered (SISS) which can be viewed as an extension of the CSS to the sub-stage level. The advantage of using a serial procedure, especially one with sub-stage-level interpolations, is demonstrated in Sec. 5.4.2.

In the final full-system test on a single-element rocket combustor, SISS is used to couple the best performing RCM in this work, CP-GNet, with a high-fidelity FOM solver across different time-step sizes and numbers of sub-stages. With a single training process based on limited provided data from a FOM with a characteristic boundary and perturbation, the integrated framework is shown to be predictive at different chamber lengths for a much longer period than the training interval. At this point, the author would like to conclude that the thesis has introduced promising model candidates to the framework, and paved their road to be coupled with a solver in the challenge of efficiently modeling complex physics for many-query scenarios in the low data regime.

6.4 Limitations and Perspectives

Although the findings of this work are mostly validated in rocket-combustion-related applications, the data-driven methods are not restricted to the underlying problem-specific equations or physics. Indeed, various equations (ranging from 2 for the viscous Burgers problem in Sec. 5.5 to 8 for the reacting injector in Sec. 4.6.1), and types of variables (primitive or conservative) have been covered in the test cases, across which the models are easily applied by modifying the sizes of the input and output layers. More limitations are observed in cases involving the prediction for a different spatio-temporal domain than the training. As classic POD-Galerkin ROMs, the non-intrusive ROMs in this work face the same restriction to a single fixed geometry through the training and prediction stages. The multi-domain framework

endows significantly more flexibility to the applications of ROMs, yet still requires a fixed geometry in the ROM sub-domain. Additionally, the multi-level autoencoder network requires a fixed time period to be predicted between different parameters. Local surrogate models, especially the CP-GNet, eliminate the dependence on the global spatio-temporal domain, and generalize to more applications. With the help of interface models such as the interface MLP, the model-solver coupling approaches in this work are also shown to generalize to different types and schemes of models and solvers.

However, a few gaps still need to be filled between the thesis and a complete and reliable realization of the multi-fidelity framework for practical industrial problems. These gaps are aligned with the suggested future work in Fig. 1.3:

1. Data requirement for generalizability. As illustrated in the study of the 2D CVRC (Sec. 5.6), even for a highly portable model such as the CP-GNet, the representativeness of the training data still has a strong impact on the predictive capabilities of the resulting model. It can be expected that even with the help of broadband characteristic perturbations in the reduced-domain training, multiple sets of training data would still be necessary to make the model generalizable to a wide range of testing configurations. To minimize the total training cost, pre-defined criteria on the data requirement need to be developed, such that the design of training simulations (e.g. number of simulations, frequencies included, snapshots per training set, etc.) can be guided. On the other hand, further improvements on the training-data efficiency of models will remain necessary.

2. Reduced-domain FOM with multiple characteristic boundaries. The characteristic training method has only been applied to reduced-domains with a single interface in this thesis. In a larger system with multiple adjacent reduced-domains, each of them will include multiple interfaces. To perform training

FOM in such a reduced-domain, multiple groups of characteristic perturbations may be applied, the interactions of which still need to be explored.

3. Long-time stability of surrogate models. Currently, most surrogate models in the thesis and related work suffer from numerical instabilities in long-time roll-out predictions. Stability improvements are commonly achieved through treatments in the training stage, such as additional regularization terms or injection of training noises, at the cost of a lower inference accuracy. The parameters for these treatments are often determined heuristically, and do not guarantee unconditional stability. In pursuit of the latter, more efforts are needed throughout the life-cycle of a model, including architecture design, training, and inference. For example, *Pan and Duraisamy* (2020) developed a stabilized Koopman operator surrogate model for nonlinear dynamics, which enforces stabilization in the lifted space.

4. Coupling with mirrored models and different FOMs. In our framework for a complete multi-injector rocket engine combustor, multiple mirrored models are coupled together with a solver of a lower fidelity. Besides implementation challenges brought by multiple interfaces, a different FOM solver in the coupled sub-domain will probably result in misaligned mesh grids, possibly coarser time-step sizes than the model, or even a different set of variables to be solved. The current coupling schemes and interface models are not prepared to deal with these mismatches.

Beyond the current target framework, in a broader landscape for the efficient, portable, and predictive modeling of physical systems, more challenges occur. A few examples are:

1. Effective training method for more complicated features. If external responses are not acoustics-dominated, and cannot be well represented by a super-imposition

of simple boundary perturbation functions, a more complicated training data generation method will be needed.

2. *A priori* error bounds and uncertainty quantification. Most recent non-intrusive models fall into deep-learning models, which are significantly less interpretable compared with intrusive ones. Even in more intuitive architectures such as the CP-GNet, exact computations are still hidden. In this situation, *a priori* error bounds and uncertainty quantification are critical for a convincing application of model outcomes.

3. Benchmark cases and metrics. As a rapidly evolving research area, researchers on data-driven models for scientific computing have not reached any tacit agreement on how to evaluate different models and methods systematically. Even the widely used flow-over-a-cylinder case has countless flavors. A few commonly recognized benchmark cases and metrics will largely improve the situation, and consequently accelerate the realization of a reliable data-driven RCM approach. On this path, a prototyping environment for ROM methods (*Wentland*, 2021) has been recently developed, which provided test cases along with an easy-to-use coding framework.

With these challenges, it is currently not possible to completely rely on data-driven models in most current applications. However, advancements in related techniques are happening at a rapid pace. The author wishes that the leap from the initial quasi-1D CVRC test case with simplified combustion models, to the final 2D CVRC case with much more complex physics, will serve as a thumbnail for a promising future of scientific modeling.

APPENDICES

APPENDIX A

Other Applications of CPNets

A.1 Closure Modeling

In many practical problems, high fidelity simulations are not affordable. Instead, computations are performed using coarse-grained models, e.g. the Large Eddy Simulation (*Moin*, 2002). In such models, the small-scale physics are unresolved, and are approximated using additional *closure* terms in the PDEs, the development of which constitutes an important area of research. In fact, even for the seemingly simple (yet richly non-linear) equation presented below, a perfect closure model is unknown. In this work, we demonstrate how CP models can be used to develop a closure model for the coarse-grained 1D viscous Burgers equation that is often used in the study of shock formation, traffic flows, and turbulent interactions, etc. For the unknown spatio-temporal field $u(x,t)$ on a spatially periodic domain $x \in [0, L]$, the original equation is given by:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0, \tag{A.1}$$

where ν is a diffusion coefficient and $u(x,0)$ is a random IC. For the present case, it is given by:

$$u(x,0) = \sum_{k=1}^{8} \sqrt{2E(k)} \sin(kx + \beta_k), \tag{A.2}$$

where for each k, $\beta_k \sim \mathcal{U}(-\pi, \pi)$, and $E(k) = \max(k, 5)^{-5/3}$. Other choices of parameters include domain length $L = 2\pi$, viscosity $\nu = 0.01$.

A 2048 mesh point high-resolution solution is generated using the Fourier-Galerkin spectral method (*Basdevant et al.*, 1986) with 4th order Runge-Kutta method for time stepping, and used as the ground truth for the fine solution. In this setting, u can be regarded as fully resolved, thus the numerical residual $r(u)$, defined in Eq. (A.3), is zero.

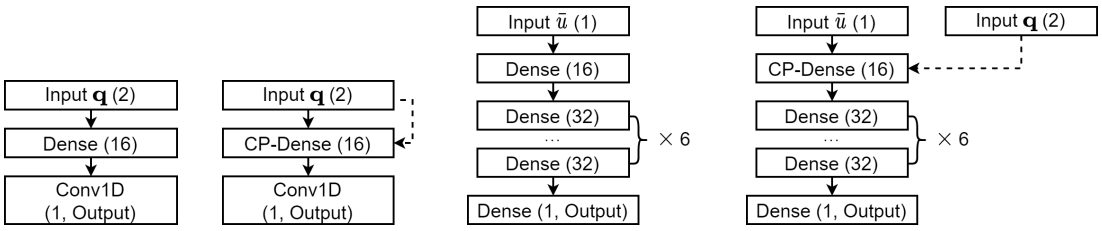
$$r(u) = -u\frac{\partial u}{\partial x} + \nu\frac{\partial^2 u}{\partial x^2} - \frac{\partial u}{\partial t}. \tag{A.3}$$

However, the same does not hold when a solution is attempted on a coarse mesh with Eq. (A.1) without any additional treatments, in which case the solution becomes inaccurate and numerically unstable, thus a closure operator $\mathcal{C}(\cdot)$ is needed to compensate for the non-zero residual. Representing the quantity on the lower resolution mesh by $\bar{u}$, the "closed" equation is:

$$\frac{\partial \bar{u}}{\partial t} + \bar{u}\frac{\partial \bar{u}}{\partial x} - \nu\frac{\partial^2 \bar{u}}{\partial x^2} + \mathcal{C} = 0. \tag{A.4}$$

In this experiment, two baseline models for $\mathcal{C}$ and their CP developments are compared. The first model is 2-layer CNN with a dense layer with ReLU activation, followed by a 1D convolution layer. This model assumes the closure term to be a function of convection term $\bar{u}\frac{\partial \bar{u}}{\partial x}$ and the diffusion term $\nu\frac{\partial^2 \bar{u}}{\partial x^2}$, and takes their concatenation $\mathbf{q} = [\bar{u}\frac{\partial \bar{u}}{\partial x}, \nu\frac{\partial^2 \bar{u}}{\partial x^2}]$ as the input. Its CP variant, CP-CNN, replaces the first layer with a CP-Dense layer that takes $\mathbf{q}$ as the parameter for its own weights. The second baseline model is a reference Data-Driven Parameterization (DDP) model (*Subel*

et al., 2021). The model takes $\mathcal{C}$ as a function of the filtered variable $\bar{u}$, which is modeled by an 8-layer MLP with swish activation. Similarly, the CP variant, CP-DDP replaces the first layer with a CP-Dense layer that takes $\mathbf{q}$ as the parameter for the weights for $\bar{u}$. The network architectures are presented in Fig. A.1.



(a) CNN (b) CP-CNN (c) DDP (d) CP-DDP

Figure A.1: Closure modeling network architectures. Solid arrow: input feature; dashed arrow: condition parameter; numbers: layer width.

Two sets of high resolution data are generated from two different ICs on a shared 2048-grid-node mesh. The low resolution solutions are obtained by applying a box-filter to each step of the high resolution solutions onto a 32-grid-node mesh. The ground truth for $\mathcal{C}$ is then computed based on the offline-filtered low resolution data as $\mathcal{C}^* = -r(\bar{u})$. Each set of data consists of 267 time steps, spanning a period of 2 s. The first 0.2 s of data for one IC is used for training.

Online testing computations are then carried out from the filtered, low resolution ICs using Eq. (A.4), with $\mathcal{C}$ computed based on the online solution at every time step. Central differencing is used for the spatial derivatives, which does not introduce additional artificial viscosity; thus, the solution without closure is naturally unstable.

x-t contours for the online computations are present in Fig. A.2 to compare the evolution of $\bar{u}$. Spatial profiles are also plotted at a few steps to provide more details. Despite a small time step $\Delta t = 0.0075$ s (CFL number< 0.5), without any closure term, the computation is numerically unstable and the error grows unbounded. The baseline CNN model is able to keep the solution stable within the period studied, and the CP-CNN improves the accuracy noticeably. The baseline DDP model is only

Table A.1: Closure model MAE. $\bar{u}$ Avg.: averaged over all steps for online prediction for $\bar{u}$; $\bar{u}$ final: for the final step of online prediction; Inf.: Unbounded cases.

	Training IC		Testing IC	
	$\bar{u}$ Avg.	$\bar{u}$ final	$\bar{u}$ Avg.	$\bar{u}$ final
CNN	0.23	0.41	0.16	0.23
CP-CNN	**0.15**	**0.21**	**0.09**	**0.13**
DDP	Inf.	Inf.	Inf.	Inf.
CP-DDP	0.42	0.89	0.3	0.41

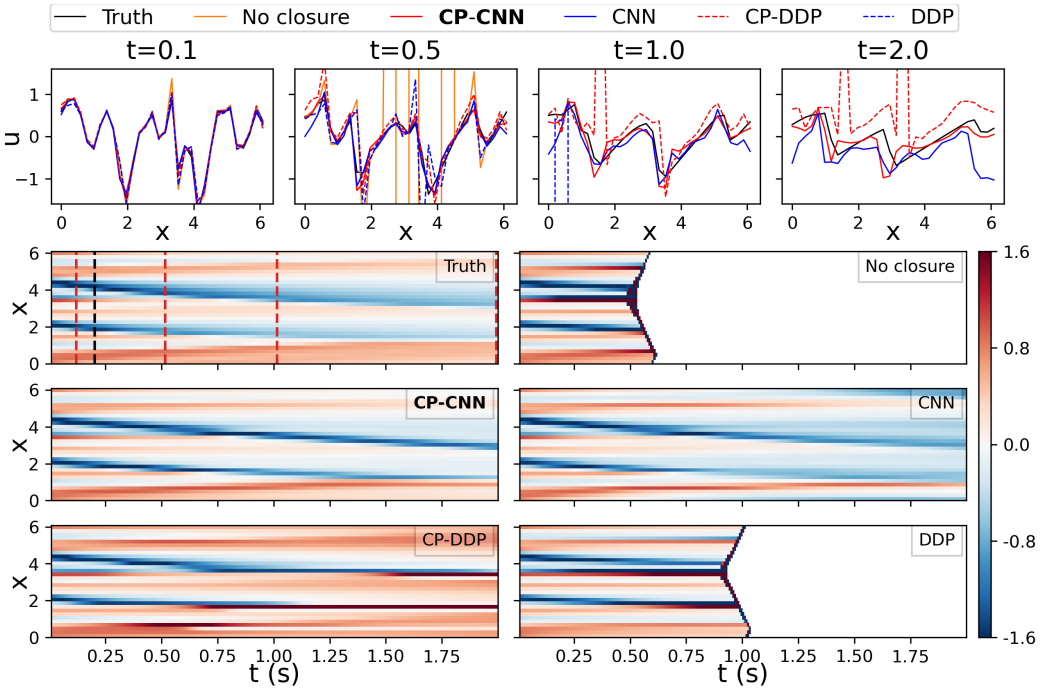
able to postpone the "blow-up" to slightly later. The solution with CP-DDP closure is bounded throughout the period. The improevments are also valid for both the unseen IC. The Mean Absolute Error (MAE) for $\bar{u}$ is provided in Table. A.1.

The comparison between CP-CNN and CNN is repeated on 4 other low resolution meshes of different sizes $n_x = \{24, 64, 128, 256\}$. The average MAE for the online computation for $\bar{u}$, and the offline single-step computation for $\mathcal{C}$ from the training IC is plotted in Fig. A.3. The CP-CNN outperforms the CNN on all meshes. Moreover, the CNN closure is unstable at the most coarse mesh, $n_x = 24$, whereas the CP-CNN remains stable.

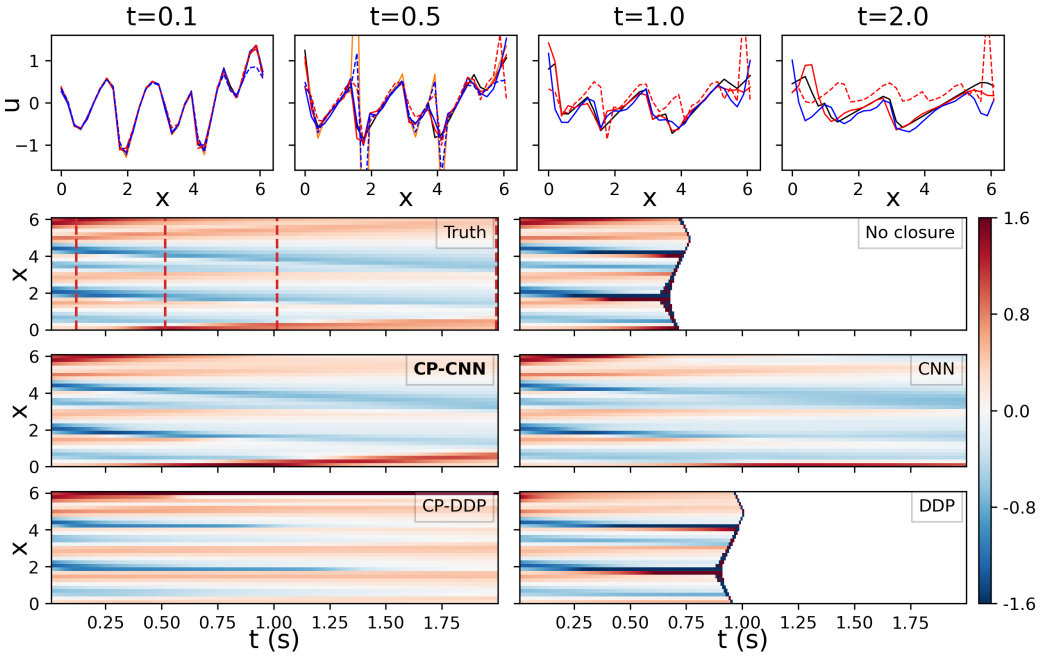
A.2 Super Resolution

In this experiment, we perform enrichment of low-resolution snapshots of turbulent flow fields. In an enrichment/super-resolution process, one inputs a low-resolution snapshot of the solution, and seeks a snapshot with better resolution. One way to achieve different resolutions on a given mesh is to use Discontinuous Galerkin (DG) projection (*Cockburn et al.*, 2012). In this method, the solution within a mesh element i is represented by coefficients $\mathbf{a}_i$ for a set of polynomial bases, of which the size is determined by the polynomial order P. The final resolution of the solution is jointly determined by P and the element width L.

More specifically, a public DNS dataset (*Del Alamo et al.*, 2004) is studied, which solves a turbulent channel flow at a friction Reynolds number $Re_\tau = \frac{u_\tau h}{2\nu} \approx 950$, where



(a) Training IC



(b) Testing IC

Figure A.2: Closure modeling results. The first $t \leq 0.2$ s for the left case is used for training, marked by the black dashed line in the first contour. The x-t contours show the evolution of $\bar{u}$. **The reference DDP model solution grows into infinity, shown as white areas in the contour**. The gaps between models are more visible in the spatial profiles at time steps marked by the red dashed lines.

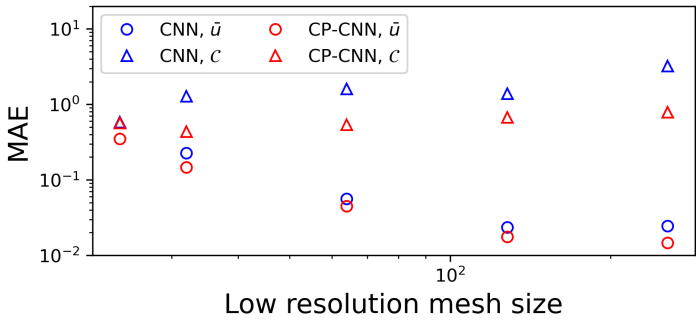


Figure A.3: Average MAE for $\bar{u}$ (online) $\mathcal{C}$ (offline) under different low resolution mesh sizes. **The CNN model blows up at $\mathbf{n_x = 24}$ and no data point is plotted.**

h is the channel height, $u_\tau = \sqrt{\tau/\rho}$ is the wall-friction velocity, defined on the averaged wall-friction τ and the density ρ. DG projection is performed for wall-parallel slices at different normalized wall-distances $z^+ = zu_\tau/\nu$, where z is the distance between the plane to the closer wall. The task is to recover high-order ($P = 3$) DG coefficients $\mathbf{a}_i^h \in \mathbb{R}^9$ for the x-velocity from lower-order ($P = 1$) ones $\mathbf{a}_i^l \in \mathbb{R}^4$. 5 snapshots are generated in total at $z^+ \in \{650, 700, 750, 800, 850\}$, as illustrated in Fig. A.4. Each snapshot spans an area of $X \times Y = 2\pi \times \pi$, and is projected onto a shared set of uniform meshes with 6 different widths $L \in \{\pi/4, \pi/8, \pi/12, \pi/16, \pi/24, \pi/32\}$, for the two studied polynomial orders $P \in \{1, 3\}$. Thus, for each z^+, 12 sets of data, each for one combination of L and P, are provided. Fig. A.4 shows a few example contours at different combinations for $z^+ = 800$. The data for $z^+ \in 700, 800$ is used for training. It should be noted that the coefficients are computed independently for each mesh element, thus the total number of training points is a few thousand, instead of 24 (which should be multiplied by the number of elements).

In this task, the baseline model is from the compact super-resolution model by *Pradhan and Duraisamy* (2021). It takes $\mathbf{a}_i^h$ as a function of two inputs. The first input is a concatenation of normalized low-order basis coefficients for i and its neighbors $N(i)$:

$$[\mathbf{a}^c]_i = [\{\mathbf{a}_j^c - \bar{\mathbf{a}}^c; j \in N(i) \cup i\}]/u_i^{\mathrm{RMS}}, \tag{A.5}$$

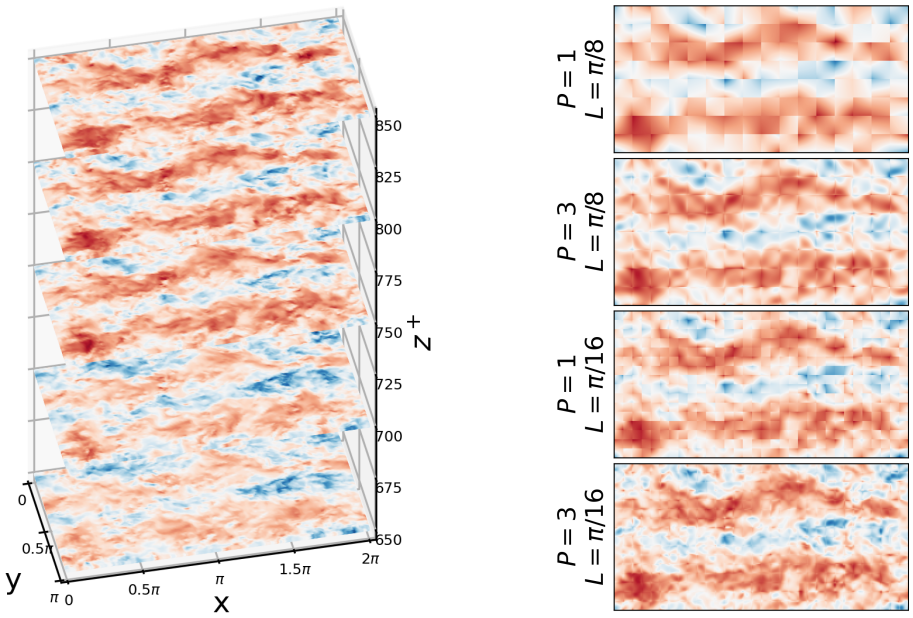
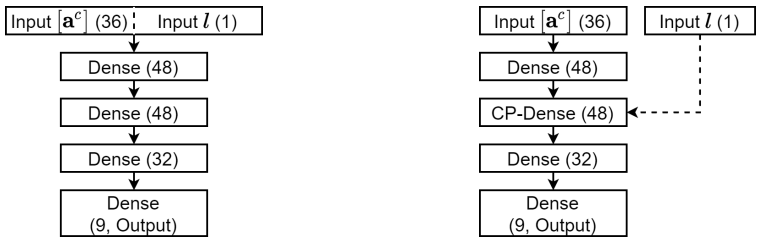


Figure A.4: Snapshots for super-resolution.



(a) MLP (b) CP-MLP

Figure A.5: Super-resolution network architectures. Solid arrow: input feature; dashed arrow: condition parameter; numbers: layer width.

where $[\{\cdot\}]$ denotes the concatenation of all elements in a set, and $\bar{\mathbf{a}}^c$ is the mean of the set. In our case, we include all immediate neighbors, including corner ones in $N(i)$, thus $[\mathbf{a}^c]_i \in \mathbb{R}^{36}$. The second input to the model is an indicator $l_i = \log(Re_i^L)$ for the loss of information in the low-order projection process. $Re_i^L = \frac{u_i^{\text{RMS}} L}{\nu}$ is the local Reynolds number. The indicator reflects that the loss is a function of the kinetic energy, measured by u_i^{RMS}, mesh resolution L, and fluid viscosity ν. Because Re_i^L can vary by orders of magnitude across elements, log scaling is used. The two inputs are first concatenated and then processed in a 4-layer MLP in the baseline model.

In contrast, the conditionally parameterized model CP-MLP processes only the first input $[\mathbf{a}^c]_i$ in the dense layers. The second dense layer is replaced by a CP-Dense layer, where the second input l_i is instead taken as a conditional parameter for the weights for the latent output of the first layer. A comparison of the model architectures are provided in Fig. A.5.

Results for two sample testing cases, $(z^+ = 650, L = \pi/4)$ and $(z^+ = 750, L = \pi/8)$ are shown in Fig. A.6. It can be observed that the CP-MLP is able to reconstruct more small scale structures compared with the MLP. The performance can be qualified by the stream-wise and span-wise energy spectra, e_x and e_y, defined as:

$$e_x(k_x) = \frac{1}{\pi} \int_{-\infty}^{\infty} \langle u(x_0, y_0) u(x_0 + x, y_0) \rangle \, e^{-ik_x x} dx, \tag{A.6}$$

$$e_y(k_y) = \frac{1}{\pi} \int_{-\infty}^{\infty} \langle u(x_0, y_0) u(x_0, y_0 + y) \rangle \, e^{-ik_y y} dy, \tag{A.7}$$

where $\langle \cdot \rangle$ denotes the average over homogeneous directions, which is the entire plane in this case. Similar to the power spectral density for a time series that describes the energy distribution over different frequencies, the energy spectra describes the energy distribution of a spatial field over different wave-numbers $k = 2\pi/\lambda$, λ being the wavelength.

e_x for different stream-wise wave numbers k_x is shown in Fig. A.6. It can be observed that for high-order projection or super-resolution, the high-wave-number spectra is much richer than that for the low-order projection. The CP-MLP plots follow the truth noticeably better than the MLP baseline, which confirms our observation from the contours. Absolute error in the integrals of energy spectra, $E_x = \int_{k_x} e_x dk_x$ and $E_y = \int_{k_y} e_y dk_y$ are computed for the 24 training and 36 testing sets and summarized in Table A.2. Both training and testing errors are reduced significantly when CP is applied.

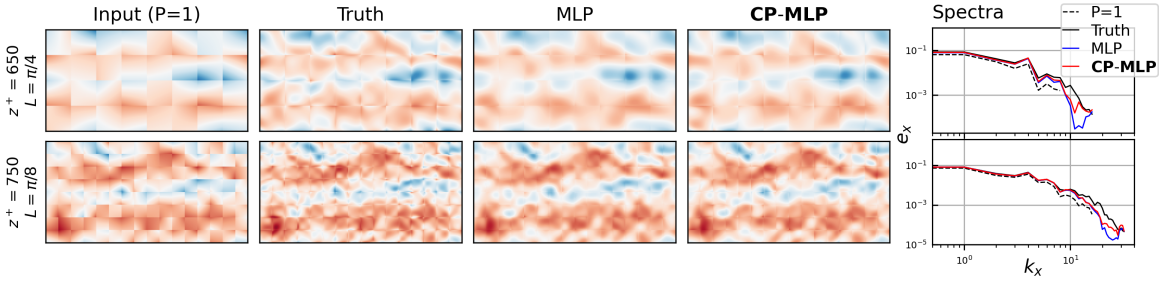


Figure A.6: Super-resolved flow field and stream-wise energy spectra e_x for example test cases $(z^+ = 650, L = \pi/4)$ and $(z^+ = 750, L = \pi/8)$. **The CP-MLP shows finer details on the edge of elements (adjacent squares)**, showing a better prediction of high-order coefficients. The observation is proved by a richer high k_x energy spectra in the right plot.

Table A.2: Average and maximum absolute errors in the integral of super-resolved energy spectra.

	Training				Testing			
	E_x Avg.	E_x Max.	E_y Avg.	E_y Max.	E_x Avg.	E_x Max.	E_y Avg.	E_y Max.
MLP	0.0145	0.0391	0.0272	0.0609	0.0098	0.0364	0.0184	0.0675
CP-MLP	**0.0120**	**0.0328**	**0.0217**	**0.0429**	**0.0081**	**0.0260**	**0.0158**	**0.0519**

APPENDIX B

Additional Results

B.1 Comparison Between CP-GNet and MeshGraphNet on the Reacting Flow

The long-training-period experiment setting from Sec. 4.6.1 is used. To apply the MeshGraphNet (MGN), one-hot labels distinguishing fluid cells and different boundary cells (inner/inlet/outlet/symmetric wall/no-slip wall), as well as the cell volumes are added to the node features. Face areas between cells are added to the edge features. Moreover, we also tested a wider version of the MGN, with the default 128-unit MLPs replaced by 256-unit ones, due to the complex reaction physics in this task. Both MeshGraphNets are trained using the same training hyper-parameters as the CP-GNet10L model from Sec. 4.6.1, and compared with the latter.

Evaluations are again performed on the representative variables $p, u, T, Y_{\mathrm{CH4}}$. The predicted flow fields are visualized in Fig. B.1. It can be seen that the CP-GNet is visually closer to the truth, especially in the phases of the probed peaks. The averaged inference time and RMSE for the normalized variables (with mean subtracted, divided by standard deviation) at different rollout steps is reported in Table B.1. The present model provides a lower RMSE throughout the prediction.

Table B.1: Averaged inference time and RMSE for reacting flow.

Model	Time/step ms	RMSE 1-step $\times 10^{-3}$	RMSE rollout-50 $\times 10^{-3}$	RMSE rollout-all $\times 10^{-3}$
CP-GNet	261	0.29	6.8	46.1
128-unit MGN	203	0.42	10.4	62.8
256-unit MGN	296	0.41	10.8	58.4

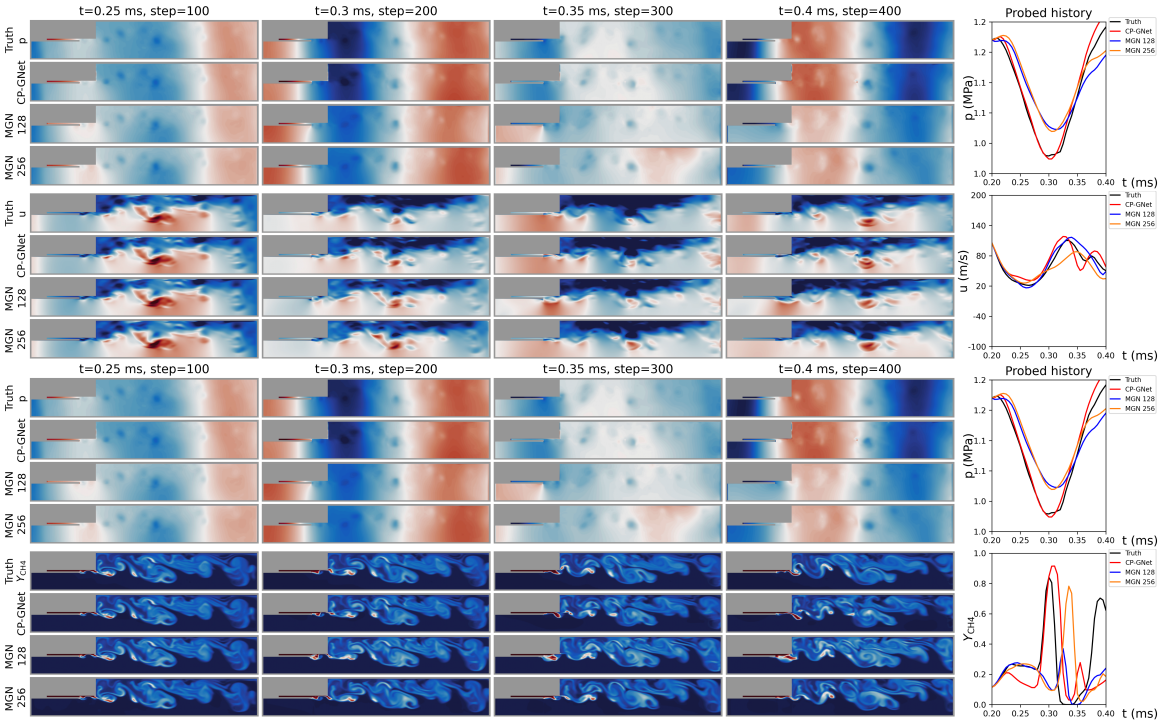


Figure B.1: Predicted reacting flow. For each variable from top to bottom: ground truth, CP-GNet, 128-unit MGN, 256-unit MGN.

B.2 Visualizations for Other Variables in 2D CVRC

As a supplement for Fig. 5.16 and 5.19, contours for the rest variables, p, u, flamelet mixture fraction and progress parameter, are provided. They further validate the conclusions in Sec. 5.6.

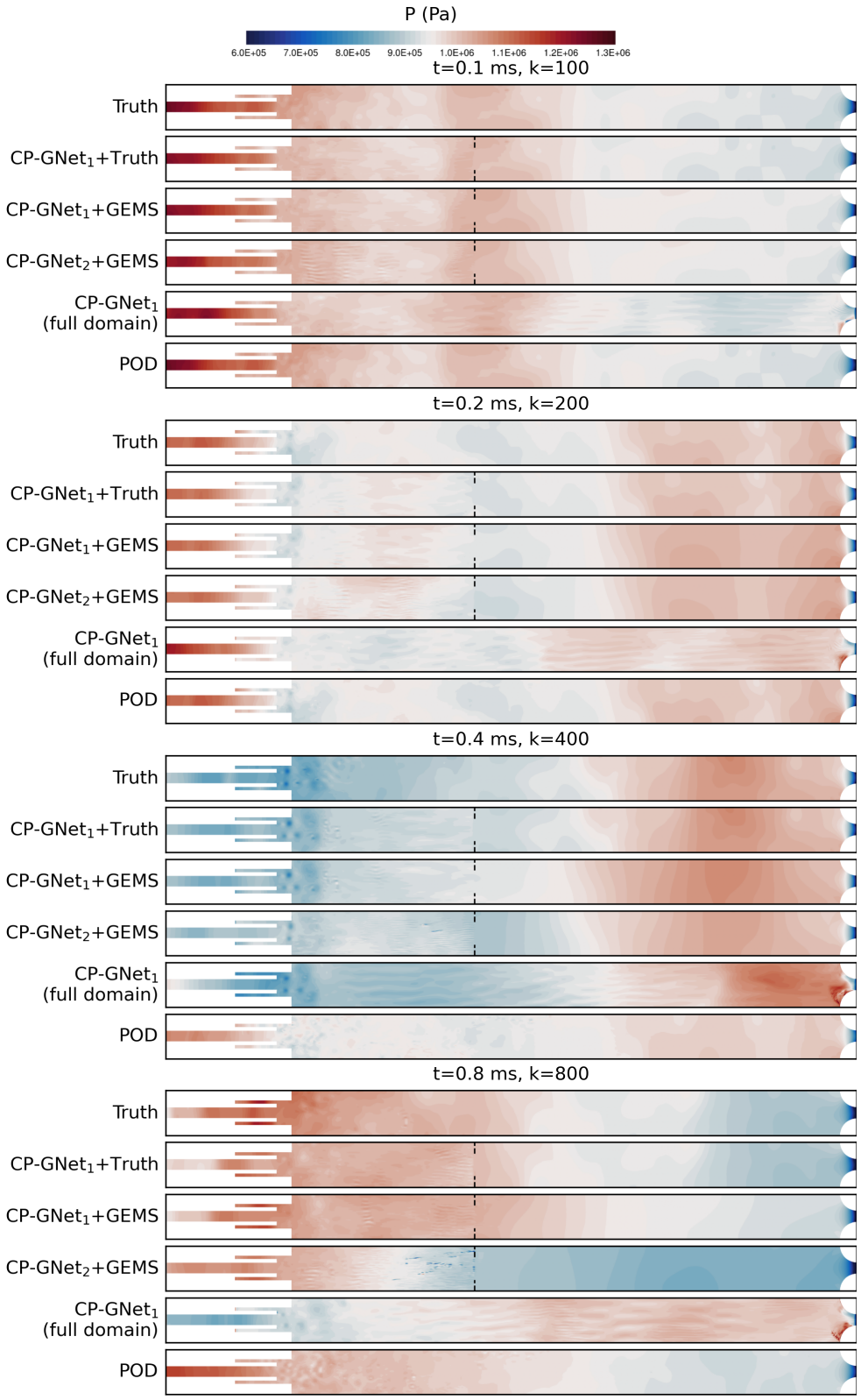


Figure B.2: Pressure contours for the longer (in-design) testing geometry.

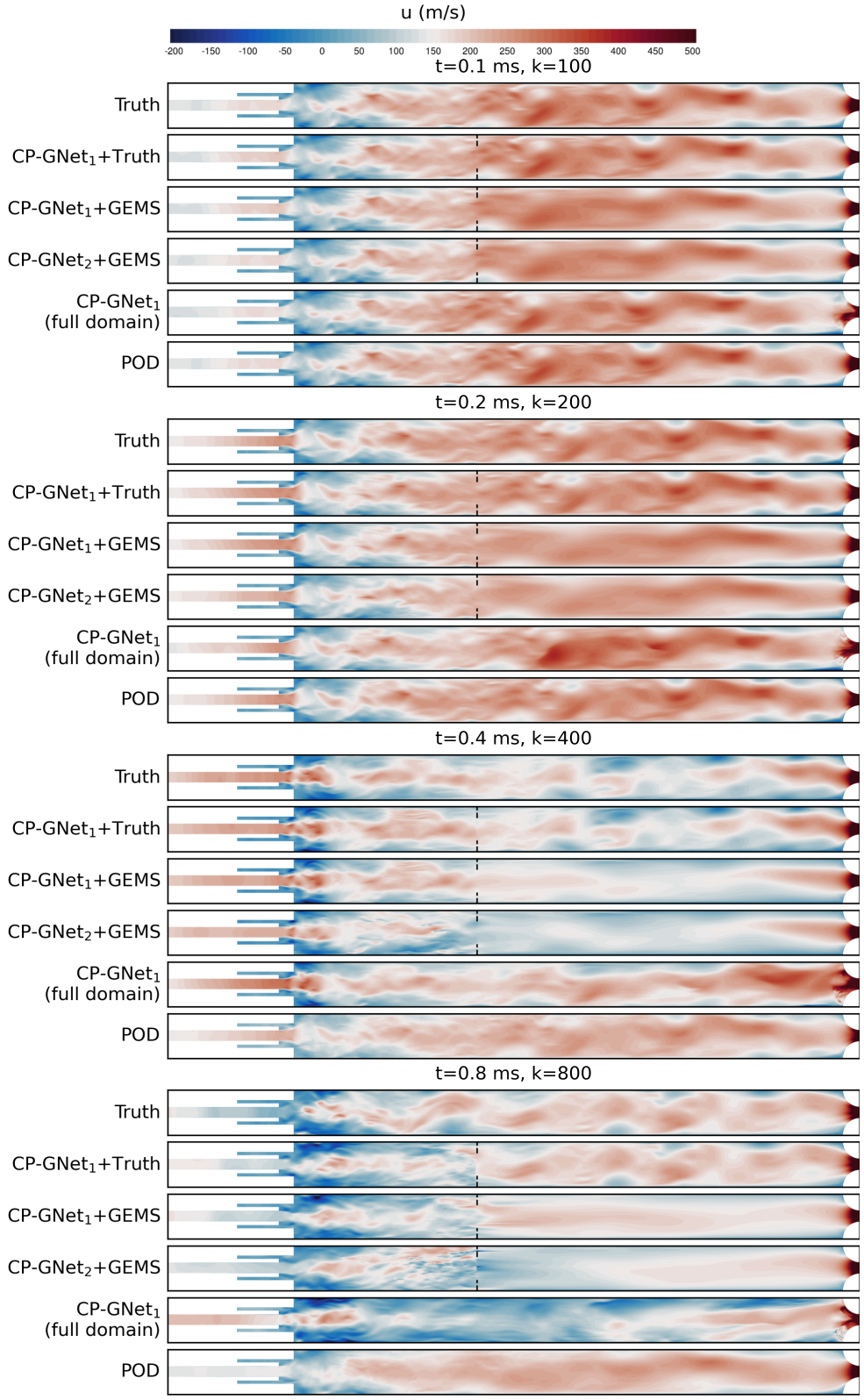


Figure B.3: x-velocity (u) contours for the longer (in-design) testing geometry.

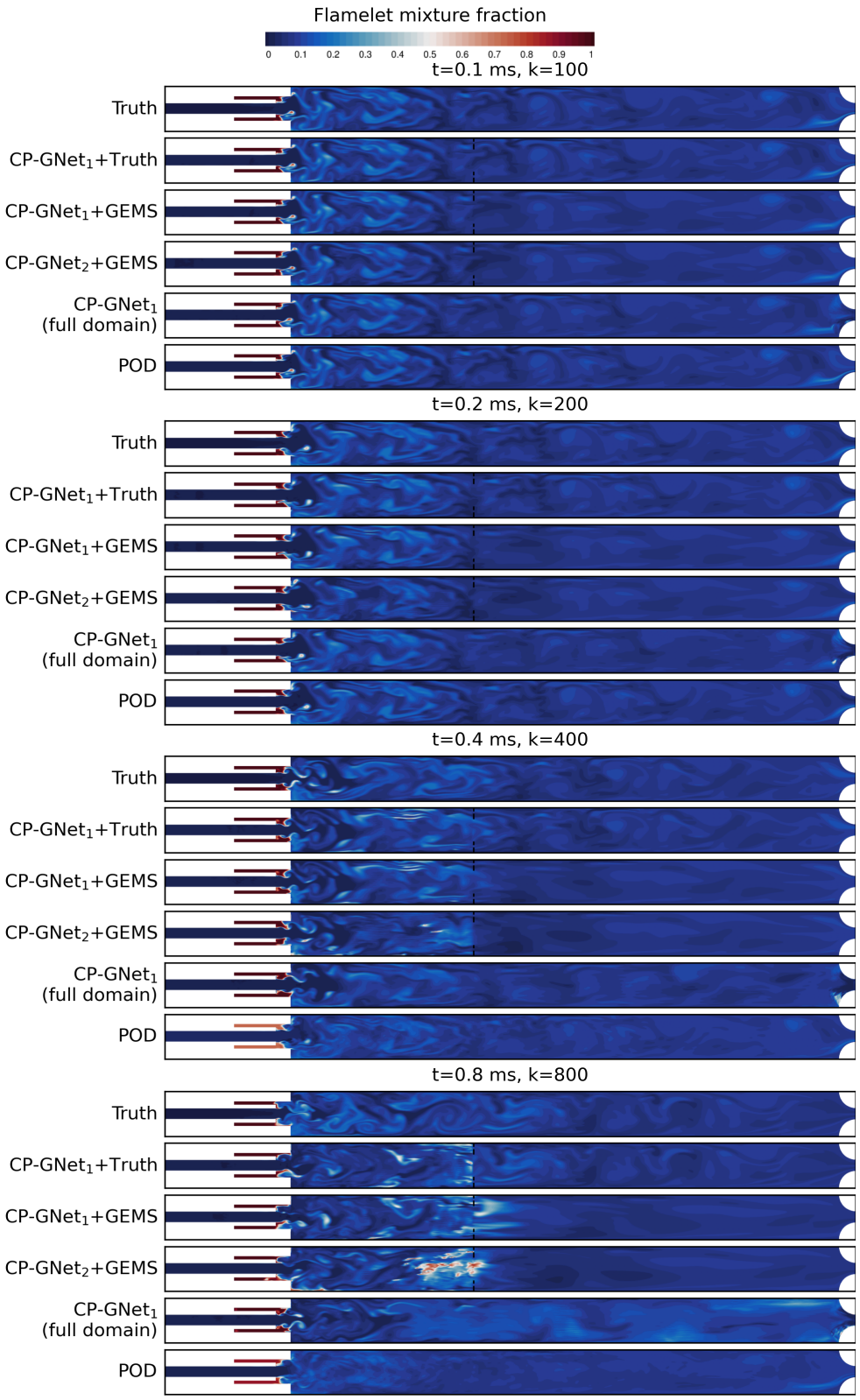


Figure B.4: Flamelet mixture fraction contours for the longer (in-design) testing geometry.

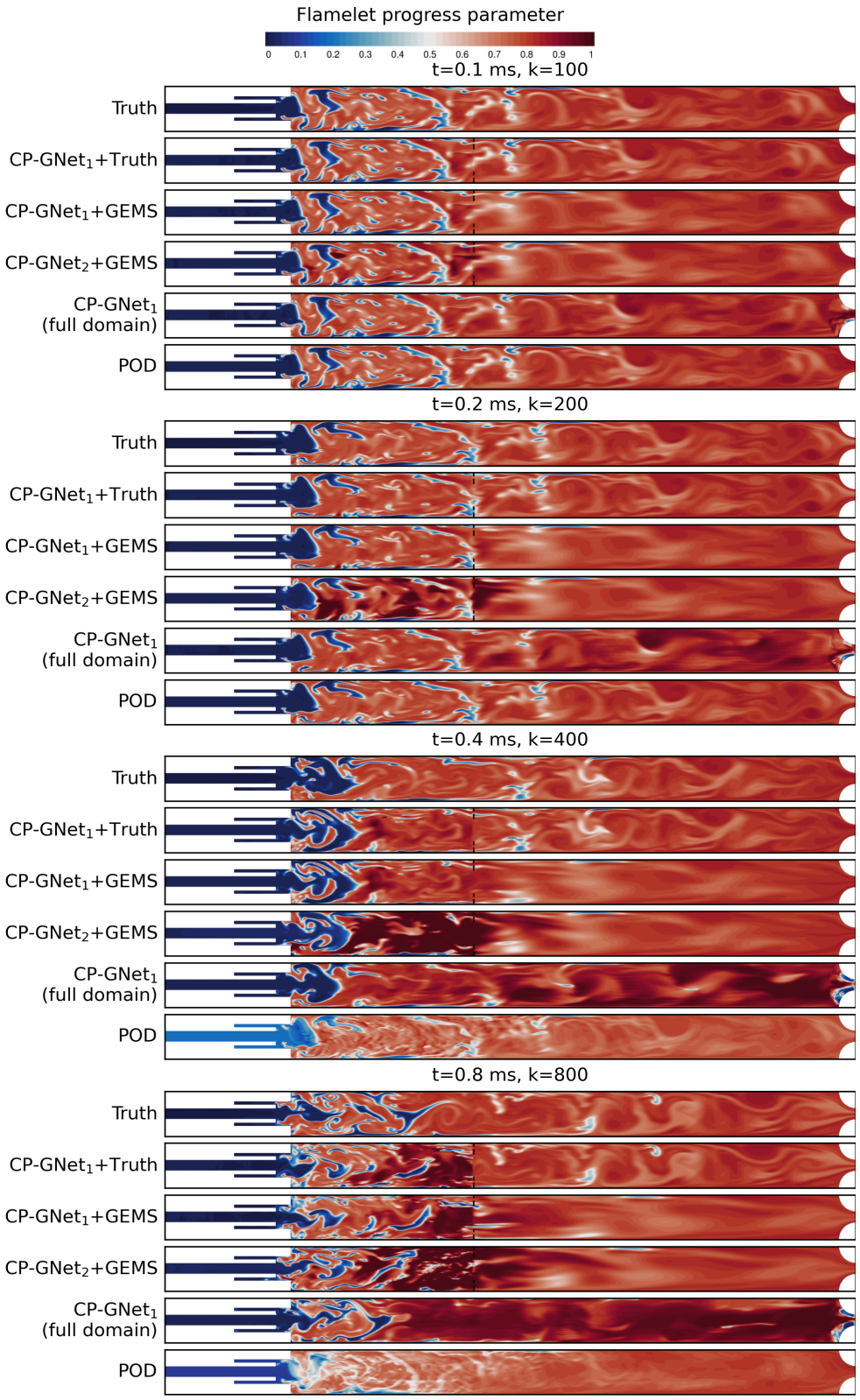


Figure B.5: Flamelet progress parameter contours for the longer (in-design) testing geometry.

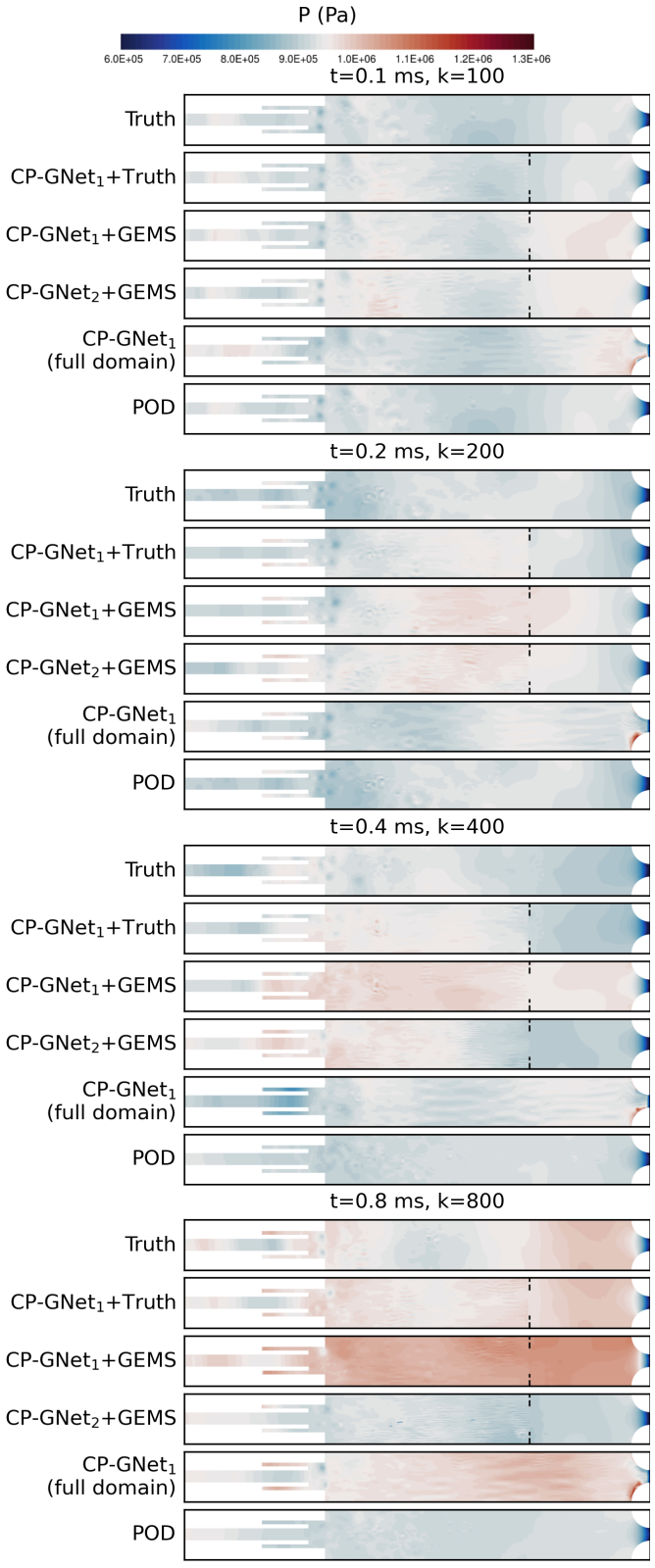


Figure B.6: Pressure contours for the shorter (off-design) testing geometry.

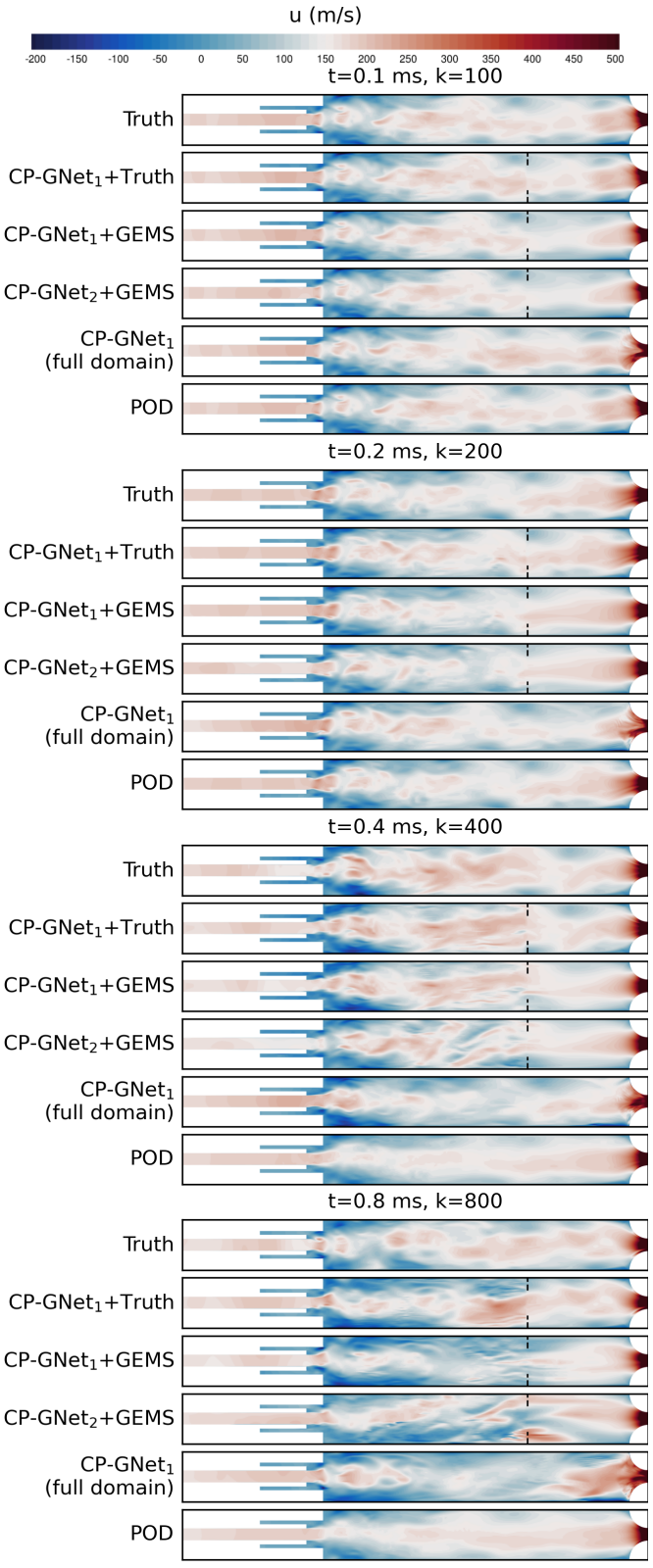


Figure B.7: x-velocity (u) contours for the shorter (off-design) testing geometry.

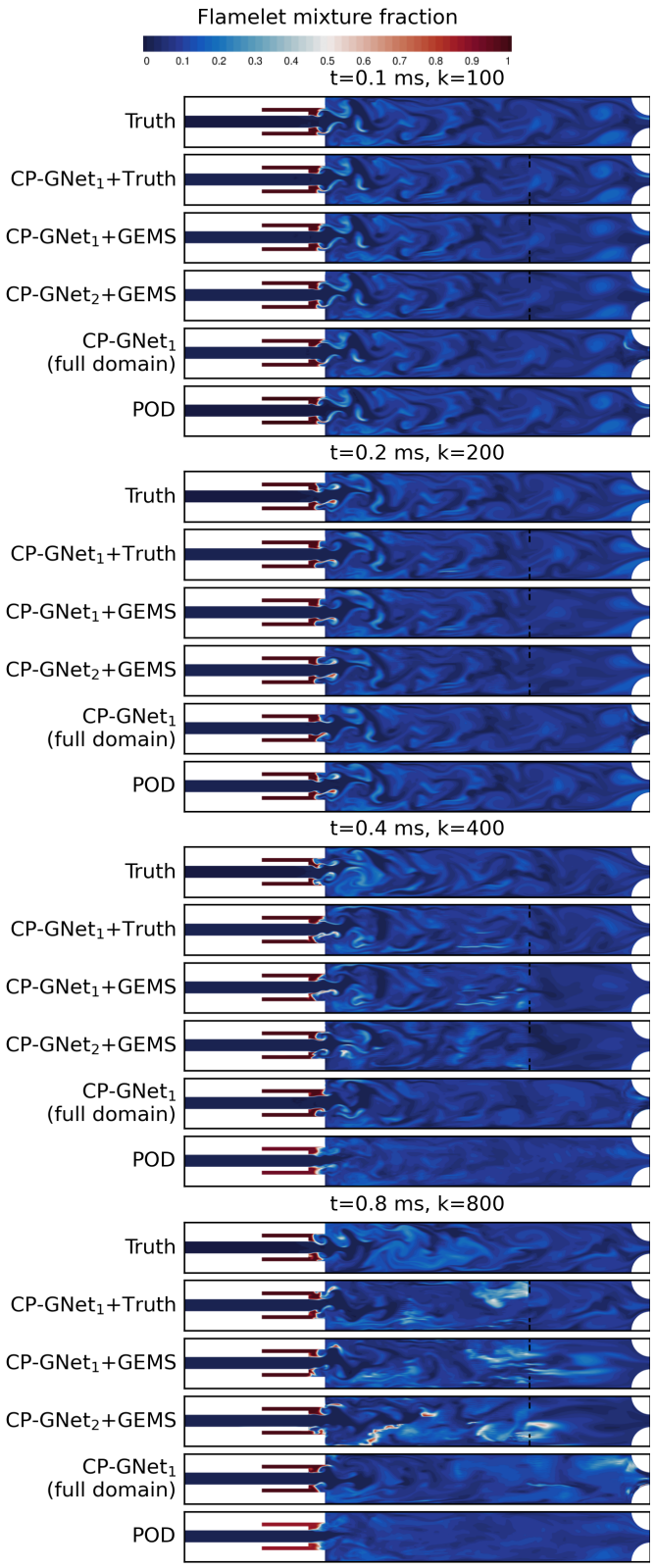


Figure B.8: Flamelet mixture fraction contours for the shorter (off-design) testing geometry.

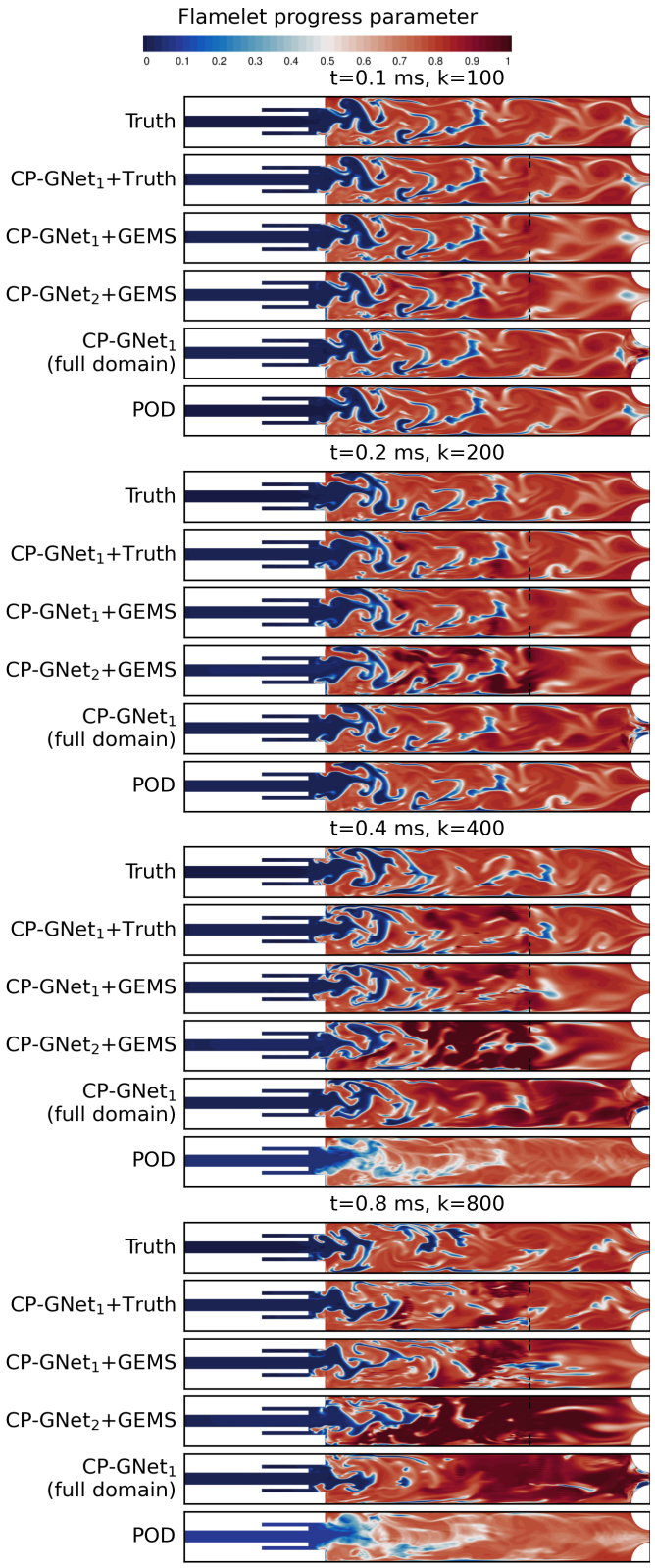


Figure B.9: Flamelet progress parameter contours for the shorter (off-design) testing geometry.

BIBLIOGRAPHY

BIBLIOGRAPHY

Amsallem, D., and C. Farhat (2011), An online method for interpolating linear parametric reduced-order models, *SIAM Journal on Scientific Computing*, *33*(5), 2169–2198.

Argaud, J., B. Bouriquet, H. Gong, Y. Maday, and O. Mula (2017), Stabilization of (g) eim in presence of measurement noise: application to nuclear reactor physics, in *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2016*, pp. 133–145, Springer.

Astrid, P. (2004), *Reduction of process simulation models: a proper orthogonal decomposition approach*, Technische Universiteit Eindhoven Eindhoven, Netherlands.

Astrid, P., S. Weiland, K. Willcox, and T. Backx (2008), Missing point estimation in models described by proper orthogonal decomposition, *IEEE Transactions on Automatic Control*, *53*(10), 2237–2251.

Ba, J. L., J. R. Kiros, and G. E. Hinton (2016), Layer normalization, *arXiv preprint arXiv:1607.06450*.

Baiges, J., R. Codina, and S. Idelsohn (2013), A domain decomposition strategy for reduced order models. application to the incompressible navier–stokes equations, *Computer Methods in Applied Mechanics and Engineering*, *267*, 23–42.

Baldwin, B., and T. Barth (1991), A one-equation turbulence transport model for high reynolds number wall-bounded flows, in *29th aerospace sciences meeting*, p. 610.

Bar-Sinai, Y., S. Hoyer, J. Hickey, and M. P. Brenner (2019), Learning data-driven discretizations for partial differential equations, *Proceedings of the National Academy of Sciences*, *116*(31), 15,344–15,349.

Basdevant, C., M. Deville, P. Haldenwang, J. Lacroix, J. Ouazzani, R. Peyret, P. Orlandi, and A. Patera (1986), Spectral and finite difference solutions of the burgers equation, *Computers & fluids*, *14*(1), 23–41.

Bathe, K., C. Nitikitpaiboon, and X. Wang (1995), A mixed displacement-based finite element formulation for acoustic fluid-structure interaction, *Computers & Structures*, *56*(2-3), 225–237.

Bathe, M., and R. Kamm (1999), A fluid-structure interaction finite element analysis of pulsatile blood flow through a compliant stenotic artery, *Journal of biomechanical engineering*, *121*(4), 361–369.

Baur, U., C. Beattie, P. Benner, and S. Gugercin (2011), Interpolatory projection methods for parameterized model reduction, *SIAM Journal on Scientific Computing*, *33*(5), 2489–2518.

Becker, S., and E. Laurien (2003), Three-dimensional numerical simulation of flow and heat transport in high-temperature nuclear reactors, *Nuclear Engineering and Design*, *222*(2-3), 189–201.

Belbute-Peres, F. d. A., T. Economon, and Z. Kolter (2020), Combining differentiable pde solvers and graph neural networks for fluid flow prediction, in *International Conference on Machine Learning*, pp. 2402–2411, PMLR.

Berger, J., S. Gasparin, M. Chhay, and N. Mendes (2016), Estimation of temperature-dependent thermal conductivity using proper generalised decomposition for building energy management, *Journal of Building Physics*, *40*(3), 235–262.

Berkooz, G., P. Holmes, and J. L. Lumley (1993), The proper orthogonal decomposition in the analysis of turbulent flows, *Annual review of fluid mechanics*, *25*(1), 539–575.

Bhatnagar, S., Y. Afshar, S. Pan, K. Duraisamy, and S. Kaushik (2019), Prediction of aerodynamic flow fields using convolutional neural networks, *Computational Mechanics*, *64*(2), 525–545.

Bose, S. T., and G. I. Park (2018), Wall-modeled large-eddy simulation for complex turbulent flows, *Annual review of fluid mechanics*, *50*, 535–561.

Brunton, S. L., J. L. Proctor, and J. N. Kutz (2016), Discovering governing equations from data by sparse identification of nonlinear dynamical systems, *Proceedings of the National Academy of Sciences*, *113*(15), 3932–3937.

Buffoni, M., H. Telib, and A. Iollo (2009), Iterative methods for model reduction by domain decomposition, *Computers & Fluids*, *38*(6), 1160–1167.

Carlberg, K., C. Bou-Mosleh, and C. Farhat (2011), Efficient non-linear model reduction via a least-squares petrov–galerkin projection and compressive tensor approximations, *International Journal for Numerical Methods in Engineering*, *86*(2), 155–181.

Carlberg, K. T., A. Jameson, M. J. Kochenderfer, J. Morton, L. Peng, and F. D. Witherden (2019), Recovering missing cfd data for high-order discretizations using deep neural networks and dynamics learning, *Journal of Computational Physics*.

Chaturantabut, S., and D. C. Sorensen (2009), Discrete empirical interpolation for nonlinear model reduction, in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pp. 4316–4321, IEEE.

Chen, W., Q. Wang, J. S. Hesthaven, and C. Zhang (2021), Physics-informed machine learning for reduced-order modeling of nonlinear problems, *Journal of Computational Physics*, *446*, 110,666.

Choi, H., and P. Moin (2012), Grid-point requirements for large eddy simulation: Chapman's estimates revisited, *Physics of fluids*, *24*(1), 011,702.

Cockburn, B., G. E. Karniadakis, and C.-W. Shu (2012), *Discontinuous Galerkin methods: theory, computation and applications*, vol. 11, Springer Science & Business Media.

Corigliano, A., M. Dossi, and S. Mariani (2015), Model order reduction and domain decomposition strategies for the solution of the dynamic elastic–plastic structural problem, *Computer Methods in Applied Mechanics and Engineering*, *290*, 127–155.

Couplet, M., C. Basdevant, and P. Sagaut (2005), Calibrated reduced-order pod-galerkin system for fluid flow modelling, *Journal of Computational Physics*, *207*(1), 192–220.

Crocco, L., J. Grey, and D. Harrje (1958), On the importance of the sensitive time lag in longitudinal high-frequency rocket combustion instability, *Jet Propulsion*, *28*(12), 841–843.

de Almeida, J. M. (2013), A basis for bounding the errors of proper generalised decomposition solutions in solid mechanics, *International Journal for Numerical Methods in Engineering*, *94*(10), 961–984.

Del Alamo, J. C., J. Jiménez, P. Zandonade, and R. D MOSER (2004), Scaling of the energy spectra of turbulent channels, *Journal of Fluid Mechanics*, *500*, 135.

DeMers, D., and G. W. Cottrell (1993), Non-linear dimensionality reduction, in *Advances in neural information processing systems*, pp. 580–587.

Dettmer, W. G., and D. Perić (2013), A new staggered scheme for fluid–structure interaction, *International Journal for Numerical Methods in Engineering*, *93*(1), 1–22.

Dissanayake, M., and N. Phan-Thien (1994), Neural-network-based approximations for solving partial differential equations, *communications in Numerical Methods in Engineering*, *10*(3), 195–201.

Domingo, P., L. Vervisch, and J. Réveillon (2005), Dns analysis of partially premixed combustion in spray and gaseous turbulent flame-bases stabilized in hot air, *Combustion and Flame*, *140*(3), 172–195.

Dong, C., C. C. Loy, K. He, and X. Tang (2015), Image super-resolution using deep convolutional networks, *IEEE transactions on pattern analysis and machine intelligence*, *38*(2), 295–307.

Dow, E., and Q. Wang (2011), Quantification of structural uncertainties in the k-w turbulence model, in *52nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 19th AIAA/ASME/AHS Adaptive Structures Conference 13t*, p. 1762.

Drmac, Z., and S. Gugercin (2016), A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions, *SIAM Journal on Scientific Computing*, *38*(2), A631–A648.

Duraisamy, K., Z. J. Zhang, and A. P. Singh (2015), New approaches in turbulence and transition modeling using data-driven techniques, in *53rd AIAA Aerospace Sciences Meeting*, p. 1284.

Erbts, P., and A. Düster (2012), Accelerated staggered coupling schemes for problems of thermoelasticity at finite strains, *Computers & Mathematics with Applications*, *64*(8), 2408–2430.

Farhat, C., and M. Lesoinne (2000), Two efficient staggered algorithms for the serial and parallel solution of three-dimensional nonlinear transient aeroelastic problems, *Computer methods in applied mechanics and engineering*, *182*(3-4), 499–515.

Farhat, C., K. G. Van der Zee, and P. Geuzaine (2006), Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity, *Computer methods in applied mechanics and engineering*, *195*(17-18), 1973–2001.

Farhat, C., A. Rallu, K. Wang, and T. Belytschko (2010), Robust and provably second-order explicit–explicit and implicit–explicit staggered time-integrators for highly non-linear compressible fluid–structure interaction problems, *International Journal for Numerical Methods in Engineering*, *84*(1), 73–107.

Felippa, C. A., and K.-C. Park (1980), Staggered transient analysis procedures for coupled mechanical systems: formulation, *Computer Methods in Applied Mechanics and Engineering*, *24*(1), 61–111.

Felippa, C. A., K.-C. Park, and C. Farhat (2001), Partitioned analysis of coupled mechanical systems, *Computer methods in applied mechanics and engineering*, *190*(24-25), 3247–3270.

Fernández, M. A., and M. Moubachir (2005), A newton method using exact jacobians for solving fluid–structure coupling, *Computers & Structures*, *83*(2-3), 127–142.

Frezzotti, M. L., F. Nasuti, C. Huang, C. Merkle, and W. E. Anderson (2015a), Parametric analysis of response function in modeling combustion instability by

a quasi-1d solver, in *6th EUROPEAN CONFERENCE FOR AEROSPACE SCIENCES*.

Frezzotti, M. L., F. Nasuti, C. Huang, C. Merkle, and W. E. Anderson (2015b), Response function modeling in the study of longitudinal combustion instability by a quasi-1d eulerian solver, in *51st AIAA/SAE/ASEE Joint Propulsion Conference*, p. 3840.

Fu, L., M. Karp, S. Bose, P. Moin, and J. Urzay (2018), Equilibrium wall-modeled les of shock-induced aerodynamic heating in hypersonic boundary layers, *Center for Turbulence Research Annual Research Briefs*, pp. 171–181.

Fuhrer, O., et al. (2018), Near-global climate simulation at 1 km resolution: establishing a performance baseline on 4888 gpus with cosmo 5.0, *Geoscientific Model Development*, *11*(4), 1665–1681.

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

Fukami, K., Y. Nabae, K. Kawai, and K. Fukagata (2019b), Synthetic turbulent inflow generator using machine learning, *Physical Review Fluids*, *4*(6), 064,603.

Gao, H., J.-X. Wang, and M. J. Zahr (2020), Non-intrusive model reduction of large-scale, nonlinear dynamical systems using deep learning, *Physica D: Nonlinear Phenomena*, *412*, 132,614.

Gao, H., L. Sun, and J.-X. Wang (2021), Phygeonet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain, *Journal of Computational Physics*, *428*, 110,079.

Gatzhammer, B. (2014), Efficient and flexible partitioned simulation of fluid-structure interactions, Ph.D. thesis, Technische Universität München.

Geneva, N., and N. Zabaras (2020), Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks, *Journal of Computational Physics*, *403*, 109,056.

Gilmer, J., S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl (2017), Neural message passing for quantum chemistry, in *International conference on machine learning*, pp. 1263–1272, PMLR.

Goc, K., S. Bose, and P. Moin (2020), Wall-modeled large eddy simulation of an aircraft in landing configuration, in *AIAA Aviation 2020 Forum*, p. 3002.

Gonzalez, F. J., and M. Balajewicz (2018), Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems, *arXiv preprint arXiv:1808.01346*.

Gou, X., W. Sun, Z. Chen, and Y. Ju (2010), A dynamic multi-timescale method for combustion modeling with detailed and reduced chemical kinetic mechanisms, *Combustion and Flame*, *157*(6), 1111–1121.

Grenda, J., S. Venkateswaran, and C. Merkle (1995), Application of computational fluid dynamics techniques to engine instability studies, *PROGRESS IN ASTRONAUTICS AND AERONAUTICS*, *169*, 503–528.

Grippo, L., and M. Sciandrone (2000), On the convergence of the block nonlinear gauss–seidel method under convex constraints, *Operations research letters*, *26*(3), 127–136.

Gu, C. (2011), Qlmor: A projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *30*(9), 1307–1320.

Guo, L., S. Ye, J. Han, H. Zheng, H. Gao, D. Z. Chen, J.-X. Wang, and C. Wang (2020), Ssr-vfd: Spatial super-resolution for vector field data analysis and visualization, in *2020 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 71–80, IEEE Computer Society.

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

Ha, D., A. Dai, and Q. V. Le (2016), Hypernetworks, *arXiv preprint arXiv:1609.09106*.

Haeseler, D., and O. J. Haidn (2017), *Russian Engine Technologies*, pp. 427–462, Springer International Publishing, Cham.

Harrje, D. T., and F. H. Reardon (1972), *Liquid propellant rocket combustion instability*, vol. 1, Scientific and Technical Information Office, National Aeronautics and Space ....

Harvazinski, M. E., C. Huang, V. Sankaran, T. W. Feldman, W. E. Anderson, C. L. Merkle, and D. G. Talley (2015), Coupling between hydrodynamics, acoustics, and heat release in a self-excited unstable combustor, *Physics of Fluids*, *27*(4), 045,102.

Harvazinski, M. E., R. Gejji, D. G. Talley, M. R. Orth, W. E. Anderson, and T. L. Pourpoint (2019), Modeling of transverse combustion instability, in *AIAA scitech 2019 forum*, p. 1732.

He, K., X. Zhang, S. Ren, and J. Sun (2016), Deep residual learning for image recognition, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hijazi, S., G. Stabile, A. Mola, and G. Rozza (2020), Data-driven pod-galerkin reduced order model for turbulent flows, *Journal of Computational Physics*, *416*, 109,513.

Hoang, C., Y. Choi, and K. Carlberg (2021), Domain-decomposition least-squares petrov–galerkin (dd-lspg) nonlinear model reduction, *Computer Methods in Applied Mechanics and Engineering*, *384*, 113,997.

Hochreiter, S., and J. Schmidhuber (1997), Lstm can solve hard long time lag problems, in *Advances in neural information processing systems*, pp. 473–479.

Holland, J. R., J. D. Baeder, and K. Duraisamy (2019), Field inversion and machine learning with embedded neural networks: Physics-consistent neural network training, in *AIAA Aviation 2019 Forum*, p. 3200.

Huang, C. (2022), Unpublished work, *unpublished work*.

Huang, C., W. E. Anderson, C. L. Merkle, and V. Sankaran (2016), Multi-fidelity framework for modeling combustion instability, *Tech. rep.*, AFRL/RQR Edwards AFB United States.

Huang, C., W. E. Anderson, and C. Merkle (2017), Multi-fidelity framework explorations for nonlinear euler equations, in *53rd AIAA/SAE/ASEE Joint Propulsion Conference*.

Huang, C., J. Xu, K. Duraisamy, and C. Merkle (2018), Exploration of reduced-order models for rocket combustion applications, in *2018 AIAA Aerospace Sciences Meeting*, p. 1183.

Huang, C., W. E. Anderson, C. L. Merkle, and V. Sankaran (2019a), Multifidelity framework for modeling combustion dynamics, *AIAA Journal*, *57*(5), 2055–2068.

Huang, C., K. Duraisamy, and C. L. Merkle (2019b), Investigations and improvement of robustness of reduced-order models of reacting flow, *AIAA Journal*, *57*(12), 5377–5389.

Huang, C., C. R. Wentland, K. Duraisamy, and C. Merkle (2021), Model reduction for multi-scale transport problems using model-form preserving least-squares projections with variable transformation, *Journal of Computational Physics*, p. 110742.

Iapichino, L., A. Quarteroni, and G. Rozza (2016), Reduced basis method and domain decomposition for elliptic problems in networks and complex parametrized geometries, *Computers & Mathematics with Applications*, *71*(1), 408–430.

Jameson, A. (1999), Re-engineering the design process through computation, *Journal of Aircraft*, *36*(1), 36–50.

Kafkas, A., and G. Lampeas (2020), Static aeroelasticity using high fidelity aerodynamics in a staggered coupled and rom scheme, *Aerospace*, *7*(11), 164.

Kerfriden, P., O. Goury, T. Rabczuk, and S. P.-A. Bordas (2013), A partitioned model order reduction approach to rationalise computational expenses in nonlinear fracture mechanics, *Computer methods in applied mechanics and engineering*, *256*, 169–188.

Kevrekidis, I. G., C. W. Gear, J. M. Hyman, P. G. Kevrekidid, O. Runborg, C. Theodoropoulos, et al. (2003), Equation-free, coarse-grained multiscale computation: Enabling mocroscopic simulators to perform system-level analysis, *Communications in Mathematical Sciences*, *1*(4), 715–762.

Khan, M. I., M. Waqas, T. Hayat, M. I. Khan, and A. Alsaedi (2017), Numerical simulation of nonlinear thermal radiation and homogeneous-heterogeneous reactions in convective flow by a variable thicked surface, *Journal of Molecular Liquids*, *246*, 259–267.

Kim, B., V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler (2019), Deep fluids: A generative network for parameterized fluid simulations, in *Computer Graphics Forum*, vol. 38, pp. 59–70, Wiley Online Library.

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

Kipf, T. N., and M. Welling (2016), Semi-supervised classification with graph convolutional networks, *arXiv preprint arXiv:1609.02907*.

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

Kramer, B., and K. E. Willcox (2019a), Balanced truncation model reduction for lifted nonlinear systems, *arXiv preprint arXiv:1907.12084*.

Kramer, B., and K. E. Willcox (2019b), Nonlinear model order reduction via lifting transformations and proper orthogonal decomposition, *AIAA Journal*, *57*(6), 2297–2307.

Kumpati, S. N., P. Kannan, et al. (1990), Identification and control of dynamical systems using neural networks, *IEEE Transactions on neural networks*, *1*(1), 4–27.

Kundrapu, M., and M. Keidar (2012), Numerical simulation of carbon arc discharge for nanoparticle synthesis, *Physics of Plasmas*, *19*(7), 073,510.

Lagaris, I. E., A. Likas, and D. I. Fotiadis (1998), Artificial neural networks for solving ordinary and partial differential equations, *IEEE transactions on neural networks*, *9*(5), 987–1000.

Landrieu, L., and M. Simonovsky (2018), Large-scale point cloud semantic segmentation with superpoint graphs, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4558–4567.

Lane, T. J., D. Shukla, K. A. Beauchamp, and V. S. Pande (2013), To milliseconds and beyond: challenges in the simulation of protein folding, *Current opinion in structural biology*, *23*(1), 58–65.

Launder, B. E., and D. B. Spalding (1983), The numerical computation of turbulent flows, in *Numerical prediction of flow, heat transfer, turbulence and combustion*, pp. 96–116, Elsevier.

Lawson, B. (2015), Rocket propulsion basics.

Ledig, C., et al. (2017), Photo-realistic single image super-resolution using a generative adversarial network, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690.

Lee, C. Y., and S. Cant (2017), Assessment of les subgrid-scale models and investigation of hydrodynamic behaviour for an axisymmetrical bluff body flow, *Flow, Turbulence and Combustion*, *98*(1), 155–176.

Lee, K., and K. T. Carlberg (2020), Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders, *Journal of Computational Physics*, *404*, 108,973.

Lee, M., and R. D. Moser (2015), Direct numerical simulation of turbulent channel flow up to, *Journal of Fluid Mechanics*, *774*, 395–415.

Li, L., K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar (2017), Hyperband: A novel bandit-based approach to hyperparameter optimization, *The Journal of Machine Learning Research*, *18*(1), 6765–6816.

Li, Y., J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba (2018), Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids, *arXiv preprint arXiv:1810.01566*.

Lilly, K. (1966), On the application of the eddy viscosity concept in the inertial subrange of turbulence, *National Center for Atmospheric Research manuscript*.

Löhner, R., C. Yang, and E. Onate (2007), Simulation of flows with violent free surface motion and moving objects using unstructured grids, *International Journal for Numerical Methods in Fluids*, *53*(8), 1315–1338.

Maday, Y., and E. M. Rønquist (2002), A reduced-basis element method, *Journal of scientific computing*, *17*(1), 447–459.

Marley, C. D., K. Duraisamy, and J. F. Driscoll (2015), Reduced order modeling of compressible flows with unsteady normal shock motion, in *51st AIAA/SAE/ASEE Joint Propulsion Conference*, p. 3988.

Márquez-Neila, P., M. Salzmann, and P. Fua (2017), Imposing hard constraints on deep networks: Promises and limitations, *arXiv preprint arXiv:1706.02025*.

Matheou, G., and D. Chung (2014), Large-eddy simulation of stratified turbulence. part ii: Application of the stretched-vortex model to the atmospheric boundary layer, *Journal of the Atmospheric Sciences*, *71*(12), 4439–4460.

Matthies, H. G., and J. Steindorf (2002), Partitioned but strongly coupled iteration schemes for nonlinear fluid–structure interaction, *Computers & structures*, *80*(27-30), 1991–1999.

Maulik, R., O. San, A. Rasheed, and P. Vedula (2019), Subgrid modelling for two-dimensional turbulence using neural networks, *Journal of Fluid Mechanics*, *858*, 122–144.

Maulik, R., A. Mohan, B. Lusch, S. Madireddy, P. Balaprakash, and D. Livescu (2020), Time-series learning of latent-space dynamics for reduced-order model closure, *Physica D: Nonlinear Phenomena*, p. 132368.

Maulik, R., B. Lusch, and P. Balaprakash (2021), Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders, *Physics of Fluids*, *33*(3), 037,106.

McQuarrie, S. A., C. Huang, and K. E. Willcox (2021), Data-driven reduced-order models via regularised operator inference for a single-injector combustion process, *Journal of the Royal Society of New Zealand*, *51*(2), 194–211.

Milano, M., and P. Koumoutsakos (2002), Neural network modeling for near wall turbulent flow, *Journal of Computational Physics*, *182*(1), 1–26.

Mohan, A., D. Daniel, M. Chertkov, and D. Livescu (2019), Compressed convolutional lstm: An efficient deep learning framework to model high fidelity 3d turbulence, *arXiv preprint arXiv:1903.00033*.

Moin, P. (2002), Advances in large eddy simulation methodology for complex flows, *International journal of heat and fluid flow*, *23*(5), 710–720.

Moin, P., and K. Mahesh (1998), Direct numerical simulation: a tool in turbulence research, *Annual review of fluid mechanics*, *30*(1), 539–578.

Moore, B. (1981), Principal component analysis in linear systems: Controllability, observability, and model reduction, *IEEE transactions on automatic control*, *26*(1), 17–32.

Murata, T., K. Fukami, and K. Fukagata (2020), Nonlinear mode decomposition with convolutional neural networks for fluid dynamics, *Journal of Fluid Mechanics*, *882*.

Narendra, K. S., and K. Parthasarathy (1992), Neural networks and dynamical systems, *International Journal of Approximate Reasoning*, *6*(2), 109–131.

Obiols-Sales, O., A. Vishnu, N. Malaya, and A. Chandramowliswharan (2020), Cfdnet: A deep learning-based accelerator for fluid simulations, in *Proceedings of the 34th ACM International Conference on Supercomputing*, pp. 1–12.

Oliver, T. A., and R. D. Moser (2011), Bayesian uncertainty quantification applied to rans turbulence models, in *Journal of Physics: Conference Series*, vol. 318, p. 042032, IOP Publishing.

Oord, A. v. d., S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu (2016), Wavenet: A generative model for raw audio, *arXiv preprint arXiv:1609.03499*.

Pan, S., and K. Duraisamy (2020), Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability, *SIAM Journal on Applied Dynamical Systems*, *19*(1), 480–509.

Parish, E. J., and K. Duraisamy (2016), A paradigm for data-driven predictive modeling using field inversion and machine learning, *Journal of Computational Physics*, *305*, 758–774.

Parish, E. J., C. R. Wentland, and K. Duraisamy (2020), The adjoint petrov–galerkin method for non-linear model reduction, *Computer Methods in Applied Mechanics and Engineering*, *365*, 112,991.

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

Peherstorfer, B. (2020), Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling, *SIAM Journal on Scientific Computing*, *42*(5), A2803–A2836.

Peherstorfer, B., and K. Willcox (2015), Online adaptive model reduction for nonlinear systems via low-rank updates, *SIAM Journal on Scientific Computing*, *37*(4), A2123–A2150.

Peherstorfer, B., Z. Drmac, and S. Gugercin (2020), Stability of discrete empirical interpolation and gappy proper orthogonal decomposition with randomized and deterministic sampling points, *SIAM Journal on Scientific Computing*, *42*(5), A2837–A2864.

Peters, N. (1988), Laminar flamelet concepts in turbulent combustion, in *Symposium (International) on Combustion*, vol. 21, pp. 1231–1250, Elsevier.

Peterson, J. S. (1989), The reduced basis method for incompressible viscous flow calculations, *SIAM Journal on Scientific and Statistical Computing*, *10*(4), 777–786.

Pfaff, T., M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia (2020), Learning mesh-based simulation with graph networks, *arXiv preprint arXiv:2010.03409*.

Piomelli, U., and E. Balaras (2002), Wall-layer models for large-eddy simulations, *Annual review of fluid mechanics*, *34*(1), 349–374.

Piperno, S., C. Farhat, and B. Larrouturou (1995), Partitioned procedures for the transient solution of coupled aroelastic problems part i: Model problem, theory and two-dimensional application, *Computer methods in applied mechanics and engineering*, *124*(1-2), 79–112.

Pope, S. B. (2004), Ten questions concerning the large-eddy simulation of turbulent flows, *New journal of Physics*, *6*(1), 35.

Pradhan, A., and K. Duraisamy (2021), Variational multi-scale super-resolution : A data-driven approach for reconstruction and predictive modeling of unresolved physics.

Pradhan, A., R. Biswas, and K. Duraisamy (2020), Super-resolution of finite element spaces using physics-informed deep learning networks for turbulent flows, *Bulletin of the American Physical Society*.

Prants, S., M. Y. Uleysky, and M. Budyansky (2011), Numerical simulation of propagation of radioactive pollution in the ocean from the fukushima dai-ichi nuclear power plant, in *Doklady Earth Sciences*, vol. 439, pp. 1179–1182, Springer.

Prud'Homme, C., D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, and G. Turinici (2001), Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods, *J. Fluids Eng.*, *124*(1), 70–80.

Qin, H. (2020), Machine learning and serving of discrete field theories, *Scientific Reports*, *10*(1), 1–15.

Ra, Y., and R. D. Reitz (2008), A reduced chemical kinetic model for ic engine combustion simulations with primary reference fuels, *Combustion and Flame*, *155*(4), 713–738.

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

Remmler, S., M. Fruman, and S. Hickel (2013), Direct numerical simulation of a breaking inertia–gravity wave, *Journal of Fluid Mechanics*, *722*, 424–436.

Roe, P. L. (1981), Approximate riemann solvers, parameter vectors, and difference schemes, *Journal of computational physics*, *43*(2), 357–372.

Roe, P. L. (1986), Characteristic-based schemes for the euler equations, *Annual review of fluid mechanics*, *18*(1), 337–365.

Rowley, C. W. (2005), Model reduction for fluids, using balanced proper orthogonal decomposition, *International Journal of Bifurcation and Chaos*, *15*(03), 997–1013.

Rowley, C. W., T. Colonius, and R. M. Murray (2004), Model reduction for compressible flows using pod and galerkin projection, *Physica D: Nonlinear Phenomena*, *189*(1-2), 115–129.

Rozza, G., D. B. P. Huynh, and A. T. Patera (2007), Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations, *Archives of Computational Methods in Engineering*, *15*(3), 1.

Rumelhart, D. E., G. E. Hinton, R. J. Williams, et al. (1988), Learning representations by back-propagating errors, *Cognitive modeling*, *5*(3), 1.

Safonov, M. G., and R. Chiang (1989), A schur method for balanced-truncation model reduction, *IEEE Transactions on Automatic Control*, *34*(7), 729–733.

Sanchez-Gonzalez, A., N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia (2018), Graph networks as learnable physics engines for inference and control, in *International Conference on Machine Learning*, pp. 4470–4479, PMLR.

Sanchez-Gonzalez, A., J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia (2020), Learning to simulate complex physics with graph networks, *arXiv preprint arXiv:2002.09405*.

Schalkwijk, J., H. J. Jonker, A. P. Siebesma, and E. Van Meijgaard (2015), Weather forecasting using gpu-based large-eddy simulations, *Bulletin of the American Meteorological Society*, *96*(5), 715–723.

Schmelzer, M., R. P. Dwight, and P. Cinnella (2020), Discovery of algebraic reynolds-stress models using sparse symbolic regression, *Flow, Turbulence and Combustion*, *104*(2), 579–603.

Schwarz, H. A. (1972), *Gesammelte mathematische abhandlungen*, vol. 260, American Mathematical Soc.

Sekar, V., Q. Jiang, C. Shu, and B. C. Khoo (2019), Fast flow field prediction over airfoils using deep learning approach, *Physics of Fluids*, *31*(5), 057,103.

Sharma, A., J. Xu, A. K. Padthe, P. P. Friedmann, and K. Duraisamy (2019), Simulation of maritime helicopter dynamics during approach to landing with time-accurate wind-over-deck, in *AIAA Scitech 2019 Forum*, p. 0861.

Sheikholeslami, M. (2017), Numerical simulation of magnetic nanofluid natural convection in porous media, *Physics Letters A*, *381*(5), 494–503.

Shroff, G., and R. Schreiber (1989), On the convergence of the cyclic jacobi method for parallel block orderings, *SIAM journal on matrix analysis and applications*, *10*(3), 326–346.

Simonovsky, M., and N. Komodakis (2017), Dynamic edge-conditioned filters in convolutional neural networks on graphs, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3693–3702.

Singh, A. P., K. Duraisamy, and S. Pan (2017a), Characterizing and improving predictive accuracy in shock-turbulent boundary layer interactions using data-driven models, in *55th AIAA Aerospace Sciences Meeting*, p. 0314.

Singh, A. P., R. Matai, A. Mishra, K. Duraisamy, and P. A. Durbin (2017b), Data-driven augmentation of turbulence models for adverse pressure gradient flows, in *23rd AIAA Computational Fluid Dynamics Conference*, p. 3626.

Singh, A. P., S. Medida, and K. Duraisamy (2017c), Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils, *AIAA journal*, *55*(7), 2215–2227.

Sirignano, J., J. F. MacArt, and J. B. Freund (2020), Dpm: A deep learning pde augmentation method with application to large-eddy simulation, *Journal of Computational Physics*, *423*, 109,811.

Slotnick, J. P., A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. J. Mavriplis (2014), Cfd vision 2030 study: a path to revolutionary computational aerosciences, *NASA Technical Reports*.

Smagorinsky, J. (1963), General circulation experiments with the primitive equations: I. the basic experiment, *Monthly weather review*, *91*(3), 99–164.

Smith, B. F., P. Bjorstad, and W. D. Gropp (1996), Parallel multilevel methods for elliptic partial differential equations, *Domain Decomposition*.

Spalart, P., and S. Allmaras (1992), A one-equation turbulence model for aerodynamic flows, in *30th aerospace sciences meeting and exhibit*, p. 439.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014), Dropout: a simple way to prevent neural networks from overfitting, *The journal of machine learning research*, *15*(1), 1929–1958.

Srivastava, V., and K. Duraisamy (2022), Generalizable physics-constrained modeling using learning and inference assisted by feature space engineering, *arXiv preprint arXiv:2103.16042*.

Stanley, K. O., D. B. D'Ambrosio, and J. Gauci (2009), A hypercube-based encoding for evolving large-scale neural networks, *Artificial life*, *15*(2), 185–212.

Subel, A., A. Chattopadhyay, Y. Guan, and P. Hassanzadeh (2021), Data-driven subgrid-scale modeling of forced burgers turbulence using deep learning with generalization to higher reynolds numbers via transfer learning, *Physics of Fluids*, *33*(3), 031,702.

Sun, L., H. Gao, S. Pan, and J.-X. Wang (2020), Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, *Computer Methods in Applied Mechanics and Engineering*, *361*, 112,732.

Swischuk, R., B. Kramer, C. Huang, and K. Willcox (2020), Learning physics-based reduced-order models for a single-injector combustion process, *AIAA Journal*, *58*(6), 2658–2672.

Tinoco, E., D. Bogue, T. Kao, N. Yu, P. Li, and D. Ball (2005), Progress toward cfd for full flight envelope, *The Aeronautical Journal*, *109*(1100), 451–460.

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

Tracey, B., K. Duraisamy, and J. Alonso (2013), Application of supervised learning to quantify uncertainties in turbulence and combustion modeling, in *51st AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*, p. 259.

Tracey, B. D., K. Duraisamy, and J. J. Alonso (2015), A machine learning strategy to assist turbulence model development, in *53rd AIAA aerospace sciences meeting*, p. 1287.

Um, K., R. Brand, P. Holl, N. Thuerey, et al. (2020), Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers, *arXiv preprint arXiv:2007.00016*.

Urbano, A., Q. Douasbin, L. Selle, G. Staffelbach, B. Cuenot, T. Schmitt, S. Ducruix, and S. Candel (2017), Study of flame response to transverse acoustic modes from the les of a 42-injector rocket engine, *Proceedings of the Combustion Institute*, *36*(2), 2633–2639.

Vallacher, R. R., S. J. Read, and A. Nowak (2017), *Computational social psychology*, Routledge.

van Zuijlen, A., and H. Bijl (2004), Implicit and explicit higher order time integration schemes for fluid-structure interaction computations, in *International Conference on Computational Science*, pp. 604–611, Springer.

Van Zuijlen, A., A. de Boer, and H. Bijl (2007), Higher-order time integration through smooth mesh deformation for 3d fluid–structure interaction simulations, *Journal of Computational Physics*, *224*(1), 414–430.

Vollant, A., G. Balarac, G. Geraci, and C. E. Corre (2014), Optimal estimator and artificial neural network as efficient tools for the subgrid-scale scalar flux modeling, in *Proceedings of the Summer Program*, p. 435, Stanford University Stanford, CA.

Wang, J.-X., J.-L. Wu, and H. Xiao (2017), Physics-informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on dns data, *Physical Review Fluids*, *2*(3), 034,603.

Wang, Q., J. S. Hesthaven, and D. Ray (2019a), Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem, *Journal of computational physics*, *384*, 289–307.

Wang, Y., Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon (2019b), Dynamic graph cnn for learning on point clouds, *Acm Transactions On Graphics (tog)*, *38*(5), 1–12.

Wentland, C. (2021), Prototyping environment for reacting flow order reduction methods (perform), *https://doi.org/10.5281/zenodo.5517532*.

Wilcox, D. C., et al. (1998), *Turbulence modeling for CFD*, vol. 2, DCW industries La Canada, CA.

Xiao, D., P. Yang, F. Fang, J. Xiang, C. C. Pain, and I. M. Navon (2016a), Non-intrusive reduced order modelling of fluid–structure interactions, *Computer Methods in Applied Mechanics and Engineering*, *303*, 35–54.

Xiao, D., P. Yang, F. Fang, J. Xiang, C. C. Pain, I. M. Navon, and M. Chen (2017), A non-intrusive reduced-order model for compressible fluid and fractured solid coupling and its application to blasting, *Journal of Computational Physics*, *330*, 221–244.

Xiao, D., F. Fang, C. E. Heaney, I. Navon, and C. Pain (2019), A domain decomposition method for the non-intrusive reduced order modelling of fluid flow, *Computer Methods in Applied Mechanics and Engineering*, *354*, 307–330.

Xiao, H., J.-L. Wu, J.-X. Wang, R. Sun, and C. Roy (2016b), Quantifying and reducing model-form uncertainties in reynolds-averaged navier–stokes simulations: A data-driven, physics-informed bayesian approach, *Journal of Computational Physics*, *324*, 115–136.

Xie, C., J. Wang, and E. Weinan (2020), Modeling subgrid-scale forces by spatial artificial neural networks in large eddy simulation of turbulence, *Physical Review Fluids*, *5*(5), 054,606.

Xingjian, S., Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo (2015), Convolutional lstm network: A machine learning approach for precipitation nowcasting, in *Advances in neural information processing systems*, pp. 802–810.

Xu, J., and K. Duraisamy (2017), Reduced-order modeling of model rocket combustors, in *53rd AIAA/SAE/ASEE Joint Propulsion Conference*, p. 4918.

Xu, J., and K. Duraisamy (2020), Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics, *Computer Methods in Applied Mechanics and Engineering*, *372*, 113,379.

Xu, J., C. Huang, and K. Duraisamy (2018), Multi-domain reduced-order modeling with sparse acceleration of combustion instability, in *2018 Joint Propulsion Conference*, p. 4680.

Xu, J., C. Huang, and K. Duraisamy (2019), Reduced-order modeling framework for combustor instabilities using truncated domain training, *AIAA Journal*, pp. 1–15.

Xu, J., A. Pradhan, and K. Duraisamy (2021), Conditionally parameterized, discretization-aware neural networks for mesh-based modeling of physical systems, *Advances in Neural Information Processing Systems*, *34*.

Yan, S., Y. Xiong, and D. Lin (2018), Spatial temporal graph convolutional networks for skeleton-based action recognition, in *Thirty-second AAAI conference on artificial intelligence*.

Yang, B., G. Bender, Q. V. Le, and J. Ngiam (2019a), Condconv: Conditionally parameterized convolutions for efficient inference, in *Advances in Neural Information Processing Systems*, pp. 1307–1318.

Yang, X., S. Zafar, J.-X. Wang, and H. Xiao (2019b), Predictive large-eddy-simulation wall modeling via physics-informed neural networks, *Physical Review Fluids*, *4*(3), 034,602.

Yilmaz, E., and B. German (2017), A convolutional neural network approach to training predictors for airfoil performance, in *18th AIAA/ISSMO multidisciplinary analysis and optimization conference*, p. 3660.

Young, V. (1995), *Liquid rocket engine combustion instability*, vol. 169, Aiaa.

Yu, B., H. Yin, and Z. Zhu (2017), Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, *arXiv preprint arXiv:1709.04875*.

Yu, Y., J. C. Sisco, S. Rosen, A. Madhav, and W. E. Anderson (2012), Spontaneous longitudinal combustion instability in a continuously-variable resonance combustor, *Journal of Propulsion and Power*, *28*(5), 876–887.

Yu, Y. C. (2009), Experimental and analytical investigations of longitudinal combustion instability in a continuously variable resonance combustor (cvrc), Ph.D. thesis, Purdue University.

Yuan, Z., C. Xie, and J. Wang (2020), Deconvolutional artificial neural network models for large eddy simulation of turbulence, *Physics of Fluids*, *32*(11), 115,106.

Yusuf, S. N. A., Y. Asako, N. A. C. Sidik, S. B. Mohamed, and W. M. A. A. Japar (2020), A short review on rans turbulence models, *CFD Letters*, *12*(11), 83–96.

Zhang, J., X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung (2018), Gaan: Gated attention networks for learning on large and spatiotemporal graphs, *arXiv preprint arXiv:1803.07294*.

Zhong, X., and X. Wang (2012), Direct numerical simulation on the receptivity, instability, and transition of hypersonic boundary layers, *Annual Review of Fluid Mechanics*, *44*, 527–561.

Zhuang, J., D. Kochkov, Y. Bar-Sinai, M. P. Brenner, and S. Hoyer (2020), Learned discretizations for passive scalar advection in a 2-d turbulent flow, *arXiv preprint arXiv:2004.05477*.

Zimmermann, R., B. Peherstorfer, and K. Willcox (2018), Geometric subspace updates with applications to online adaptive nonlinear model reduction, *SIAM Journal on Matrix Analysis and Applications*, *39*(1), 234–261.