Scalable and Predictive Model Order Reduction for Reacting Flow Systems

by

Nicholas Anthiah Wolfgang Arnold-Medabalimi

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering and Scientific Computing)
in the University of Michigan
2023

Doctoral Committee:

Professor Karthik Duraisamy, Co-Chair
Assistant Professor Cheng Huang, Co-Chair
Associate Professor Mirko Gamba
Associate Professor Eric Johnsen

Nicholas Arnold-Medabalimi

narnoldm@umich.edu

ORCID iD: 0000-0003-2810-7996

Dedicated to my parents, who have always pushed me to improve and always been there in times of need

Acknowledgements

I have had the fortune to spend much of my young adult life at the University of Michigan. The pursuit of self-improvement as well as contributing to scientific projects, would not be possible without an unmeasurable level of support from many people.

Professor Karthik Duraisamy has been an incredibly understanding mentor over my longer-than-average tenure. He has been a great support in helping me identify my weakness as well as flexible enough to let my thesis take some interesting turns. Most importantly, beyond his scientific insight, is his kindness and understanding of Ph.D. students' challenges and that no two academic journeys are the same.

Professor Cheng Huang has also been an incredible guide into the complex world of computational combustion. He probably has the record for the most Slack messages sent, from asking to understand some detail of the GEMS solver to discussing some exciting ideas or asking about new simulation results. He arrived at UM at the perfect time when I was entering my more research-focused years and supercharged the results of this thesis.

I would also like to thank Professor Mirko Gamba and Professor Eric Johnsen for being a part of my committee and for their input to improve the final product of my time here. Their insight has been critical in expanding my horizons as well as improving the accessibility of my work.

As a result of my time here, I have made many friendships that I would be remiss not to mention. I can't emphasize enough how the casual conversation about technical problems helped make so many research efforts click behind the scenes. I apologize for any missing names; they know who they are.

First among equals are the amazing members of CASLAB. I arrived when the group was still small and growing, and I have met and worked with a wonderful group of people,

each with their own expertise. These include Anand, Eric, Ayoub, Shaowu, Jiayang, Vishal, Aniruddhe, Chris, James, Bernardo, Christian, Jasmin, Sahil, Daisuke, and Niloy. Additionally, the Post-docs and research fellows have provided valuable scientific and practical insight. These include Asitav, Danny, Yaser, Adam, Raj, Elnaz, and Mehdi.

I want to make some specific acknowledgments. First to Anand, Helen, and Eric, who, when I arrived unsure, helped me settle and were invaluable during my first few years as office mates. Next up are Vishal and Ayoub. Both of them are geniuses in their own right, but more importantly, were calming influences who understood that research is not a linear path. I will never forget the time spent hanging out with them under planned or unplanned circumstances. Last is Chris, whose work has been most directly tied to my own. Combined with Cheng and Karthik, he was the most helpful resource in getting to the bottom of the nitty-gritty details of ROMs. His attention to detail certainly made me take a second look at things I otherwise would have missed, and I doubt I would have graduated without his help. After the departure of some older members, he also proved to be the best person to help wrangle people for social events. I hope the remaining members pick up that torch to maintain the fun within CASLAB.

Within my cohort, I cannot imagine my time here without Fabian and Sam. From grinding out prelim studies to trips to Chicago and Toronto to ritual dinners, my time here would have been far more boring without their presence.

Within the broader aerospace community, some names come to mind, from hanging out at lunch to coming to my board game nights, Krystal, Brittany, Miles, Logan, Rohan, Jake, Robin, and Yuki. Outside of the department, I was fortunate to be a part of the Rackham student government. I have met many people that have become friends even after, including but not limited to Veronica, Lucca, Alyssa, Daniel, Vinodh, Brittany, Raz, and Marshall.

I would be remiss not to mention a number of people that have been wonderful supportive friends even before my departure to Michigan. These include friends from UMD, including Platon, Andrew, Hannah, Will, Julian, Tim, David, Kevin, and many others. Last but not least, friends from the long haul of high school, Delilah, Sylvester,

and Cassie from the quartet.

Finally, I have to thank my family. My parents, John and Dagmar, have been nothing but supportive of my pursuit of my Ph.D. They have ensured that I can focus entirely on my research efforts, are always available to talk, and are a constant source of strength with their presence. I also must thank the numerous other family members who have been supportive during these challenging years and for being a rest stop on my road trips home. Hopefully, with the conclusion of my Ph.D. studies, I can spend more time with them.

The National Science Foundation has supported most of this work through grant CMMI 1634709, "A Diagnostic Modeling Methodology for Dual Retrospective Cost Adaptive Control of Complex Systems". This work was also supported through the US Air Force under the Center of Excellence grant FA9550-17-1-0195, titled "Multi-Fidelity Modeling of Rocket Combustor Dynamics". Additional computational resources were provided through the Department of Defense High-Performance Computing Modernization Project (HPC-MP).

Table of Contents

List of Tables

List of Figures

Abstract

Accurate, efficient prediction of reacting flow systems is challenging due to stiff reaction kinetics, significant disparity in spatiotemporal scales, and multi-physics interactions. Predictive tools for modeling these problems involve large-scale simulations in the form of prohibitively expensive direct numerical simulations (DNS) or slightly less expensive large-eddy simulations (LES), which can take weeks or months to run. Industry design cycle analysis requires rapid turnaround in minutes or hours, making them unsuitable for practical use. A grand challenge is, therefore, to inherit the predictive capabilities of a highly complex large-scale computation at a significantly reduced cost.

Projection-based reduced-order models - which aim for mathematically formal complexity reduction without sacrificing physical fidelity - have increased in popularity over the past two decades. These techniques have mostly been demonstrated to be successful on relatively simple problems. This thesis aims to make strides in utilizing projection-based reduced-order modeling on reacting flow systems, focusing on accuracy and scalability.

In the application of ROMs for larger-scale problems, it becomes clear that basic linear algebra pre-processing operations of large dense matrices present a significant hurdle. To better enable the development of large-scale ROMs, a software tool PLATFORM (Parallel Linear Algebra Tool FOr Reduced Modeling), has been developed to address these challenges. In addition to enabling the required distributed linear algebra, PLATFORM uses efficient I/O strategies to reduce the processing time of large data sets. This tool is ubiquitous throughout this work and critical for ROM development shown in this and other collaborative works.

An LES study of a Gas-Turbine Model Combustor (GTMC) is conducted using a

flamelet-based turbulent combustion model for two operating conditions. These cases are quantitatively compared with experimental data and show good agreement. These simulations are used as a testbed for reduced-order model development. The highly chaotic nature of the GTMC system makes static basis methods unsuitable for any truly predictive or parametric ROM. Adaptive basis techniques are applied to mitigate this shortcoming by updating the projection sub-space as the dynamics evolve. This update is guided by the residual of the full order model operator, applied to the GTMC, and shown to successfully predict future state dynamics with very few training data snapshots. The significant reduction in offline training requirements and improved accuracy are critical components to enable the application of these ROMs in a design environment.

Next, an adaptive sampling method is used to achieve computational efficiency in conjunction with the predictive capability. This method is shown to maintain accuracy in predictive tasks while significantly reducing computational costs. However, online adaptation imposes a significant challenge in parallel load balancing, which limits scalability. A framework is proposed where the sampled ROM mesh points are dynamically distributed among MPI processes during runtime. This redistribution introduces a trade-off between the cost of load balancing and the savings achieved during the sampling iteration. The framework is demonstrated on reacting flow benchmarks and quantifies the improved computational speed-up and predictive capability.

These advances in ROM methods show that the grand challenge of truly predictive and scalable ROMs for complex problems is within reach. This work makes strides in applying these methods to large-scale problems and addressing practical challenges in a high-performance computing environment. Based on the achieved efficiency and predictive capability, the author believes this work will inform and assist in developing future production-level full and reduced-order solvers.

Chapter 1

Background and Motivation

1.1 Global Energy Environment

In the last ten years, there have been strides in revolutionizing the energy sources and systems used to power everyday life. Driven by efforts to mitigate climate change, electrifying traditionally chemically powered systems offers additional conveniences. Renewable sources have made strong inroads in power grid infrastructure and transportation, where conventional combustion systems are dominant [1].

1.1.1 Power Generation

As seen in Fig. 1.1 from the U.S. Energy Information Administration, renewable sources compose a significant portion of energy generation compared to one decade ago. The increase in this component is not expected to eliminate the role of traditional combustion-based systems. Unlike fossil fuel-based energy systems, wind and solar (the major components of renewable energy sources) do not provide constant baseline output. As renewable sources increase in capacity, traditional systems will be needed to maintain grid stability. These sources provide large power output that highly dependent on the time of day Fig 1.2.

As a result, modern combustion power systems must operate in an increasingly dynamic regime as the power grid demands. This type of operation manifests a variety of problems from a combustion dynamics perspective. Traditionally, a base-load natural gas power plant would operate at a consistent energy output. This single design point

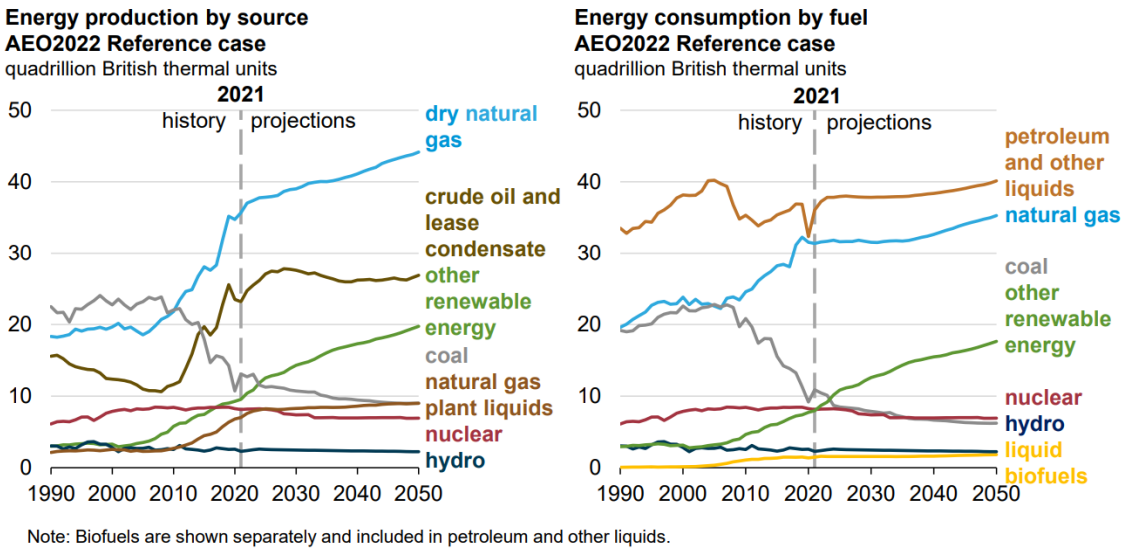


Figure 1.1: Energy production and consumption projections [1].

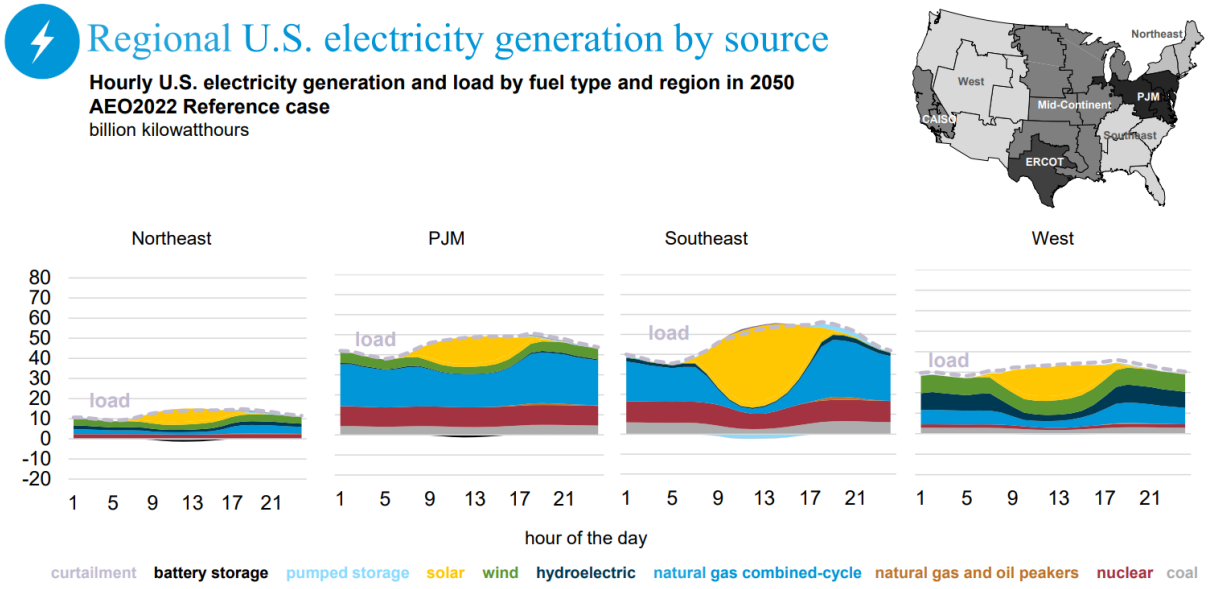


Figure 1.2: Daily power load broken down by fuel type and region [1].

significantly reduces the development cost as the combustor geometry and characteristics can be extensively validated and problems are mitigated.

Traditional Combustion-based power generation systems can have trouble adapting to this new energy environment. Older power generators can encounter instabilities or perform at significantly diminished efficiencies if operated at lower energy levels. Operations at this level can dramatically reduce the life span of a system OR make the system non-compliant with emission standards. Finally, while engineers are most concerned with the physical design of these systems, there is an important aside from an economic point of view. Older systems in the current power grid continuously monitor the cost of operation and fuel costs vs. the value of providing power to the grid. The introduction of renewable power sources into the grid system has made this value comparison extremely volatile daily and even hourly. The operators of these systems will turn off the system during unprofitable times and restart during profitable windows. These combined make the operation of a more dynamically capable power system more desirable.

Emissions

In addition to the dynamic operation challenges, emission considerations are becoming more substantial in combustor designs. Efficient combustion systems are pushed to operate in an increasingly fuel-lean environment. This operational paradigm significantly increases the probability of undesirable combustor pressure oscillations. These oscillations are difficult to predict as their inherent non-linearity means that computational tools are either incapable or extremely expensive. These combustor pressure oscillations must be identified during the design stage of new products, as the testing and manufacturing of new combustors can be extremely expensive. Therefore, the ultimate goal must be developing efficient computational design tools which can help identify these physical effects early. Predicting the onset of these instabilities has proven to be an extreme challenge over the last two decades. However, because of recent advances in computational technology, various computational tools ranging from high-fidelity simulations to significantly reduced-fidelity models have shown promise in predicting these phenomena. In section

1.2, we will review the combustion instability phenomena from a physical perspective. In section 1.4, we will review some of the different computational methods used to predict these phenomena.

1.1.2 Transportation

Unlike power generation systems, intermediate and long-range transport aircraft and launch vehicle energy systems are significantly more restricted to traditional chemical-based methods. The massive energy density differential between hydrocarbons and the current application-ready battery continues to limit the electrification of these transport systems. The energy density of jet fuel is 45 MJ/kg compared with the 0.7 MJ/kg lithium-ion battery. Even accounting for the not insignificant efficiency loss in combustion system vs. electric motors retrofitting any existing aircraft or launch vehicle would render it incapable of flight and take up all usable cargo space. However, these systems are subject to the same environmental and economic desirability as stationary power generation systems. For aircraft engines, not only are emissions tightly regulated, incremental improvement is expected every generation. Unlike static systems, the weight constraints placed on aircraft engines preclude the use of large filtering systems making meeting emission standards more challenging. Flight systems are thus constrained by their fundamental weight requirements, combustion chamber emission, and efficiency characteristics. Reaching higher efficiencies and reducing fuel waste has driven them to operate at increasingly fuel-lean regimes. These regimes put the system at greater risk for instabilities, negatively affecting the operation and device longevity. As a result, each new engine goes through significant testing and design stages to ensure stability, performance, and compliance with regulations.

Ultimately, traditional combustion systems have never been more heavily constrained and optimized. Manufacturers must go through significantly more iterations of design, testing, and analysis for each product. With the addition of transient and multiple operating conditions, conventional testing and analysis methods have become too expensive. As a result, new tools are desired to accelerate and assist in these developments.

1.2 Combustion Instability

Combustion instability (or thermoacoustic instability) has remained a significant concern in combustion device design. Characterized by large-amplitude pressure oscillations, this phenomenon can lead to catastrophic device failure. The primary mechanism of combustion instability can be attributed to the interactions between flow dynamics, acoustics, and chemical reactions. The observation of this phenomenon was first documented by Mallard and Le Chatelier [7]. Lord Rayleigh [8] proposed an elegant explanation for combustion instability based on the phase relationship between unsteady acoustics and chemical reactions (or, more precisely, the pressure and heat release fluctuations). However, in real combustion devices, combustion instability can be influenced by many factors, including geometric details, operating conditions, and reaction chemistry. Each of these components introduces additional challenges in developing predictive models, even from a high-fidelity standpoint.

General Heat Driven Instability

The general concept of the Rayleigh criteria [8] was further quantified elegantly by Culick [9, 10]. This derivation was similarly arrived at by other studies [11]; however, the general analogy compares the sound created by an oscillating heat source to an oscillating piston.

The reader can conceptualize the analogy as such; Consider a mass of gas near a wall. As the wall is instantaneously heated, the gas expands in a propagation away from the wall. If any pressure perturbations interact with the expansion front. In the analogy, the expansion front can be thought of as a piston head advancing in space. Thus in systems with in-phase pressure and heat oscillations, amplification will occur. While not precisely required, amplification is observed within a $\pm 90^{\circ}$ phase angle, with damping observed at a greater phase angle. The Rayleigh criterion describes this phase relationship:

$$\int \int \int_{\Omega} pqd\Omega > 0 \tag{1.1}$$

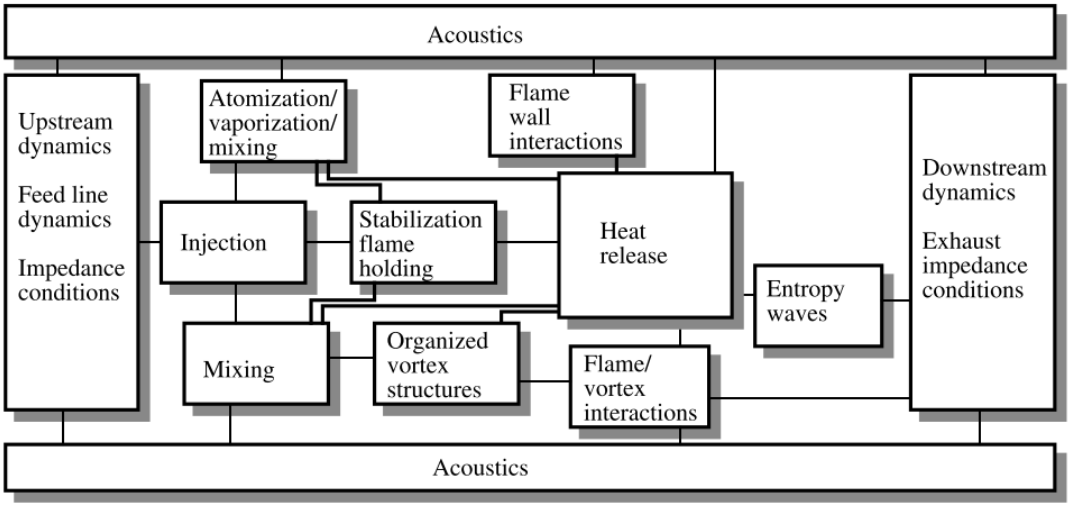


Figure 1.3: Components of combustion instability [2].

For practical combustors, combustion instabilities are not so easily quantified. The phase relationship between heat release and pressure is not analytically computable, and the criteria can only be used to post-process experimental or computation results. The potential fluctuations for a given combustion system are governed by the nonlinear interactions between acoustic phenomena, hydrodynamic effects, and chemical kinetics (Fig. 1.3). From a qualitative perspective, one can imagine a premixed combustor. As a pressure wave interacts with the inlet, the reactants will be restricted and released at once. This release and the subsequent reaction will generate a more significant pressure wave propagating through the combustion chamber. Based on the combustor geometry, these waves can interact with downstream features and constructively interfere with newly generated perturbation, increasing the overall perturbation magnitude. If this coupled interaction between the reaction, acoustics, and hydrodynamics continues in phase, as described by Rayleigh, it can lead to considerable pressure oscillations, damaging and potentially catastrophic for operations.

Combustion instability occurs when the naturally resonant characteristic time of the flow is commensurate with the characteristic time of the combustion process. The acoustic waves typical of the configuration are the feedback process for amplifying these instabilities. These waves transmit downstream effects back upstream in a feedback process. While acoustic perturbations are the primary feedback mechanism, entropy waves, and

vorticity transport can play important roles. These unsteady elements are readily reflected from downstream features back upstream. Ultimately the source of these unstable feedback mechanisms is most commonly due to one or more of the following processes [12];

1. Unsteady strain rate
2. Flame/vortex interaction
3. Acoustic/flame coupling
4. Boundary interaction
5. Equivalence ratio non-uniformity

1.3 Gas Turbine Model Combustors (GTMCs)

Section 1.1 described energy and transportation-based energy systems requirements. These considerations economically incentivize traditional gas-turbine energy systems to have the capability to operate at specific power levels and quickly adjust output to satisfy different power grid demands.

To accommodate these requirements, namely, improved efficiency, reduced emissions, and operational flexibility, gas turbine combustors are increasingly designed to operate at fuel-lean conditions, which increases susceptibility to undesirable phenomena, most notably combustion instability. The coupling between chemical reactions and acoustic waves can significantly affect device performance. Previous designs have leveraged physical adjustments to promote passive stability [13] and active control [14] at the design point to dampen these harmful effects. Still, dynamic operation adds the extra challenge of maintaining stability at multiple operating points.

As aerospace and power generation industry applications began to significantly increase requirements for their gas turbine burner power and efficiency, a greater understanding of the underlying physics was desired. To elucidate these underlying phenomena, laboratory-scale burners have been developed. These burners operate at kilowatt energy densities and feature significantly simplified geometry and improved interrogation access.

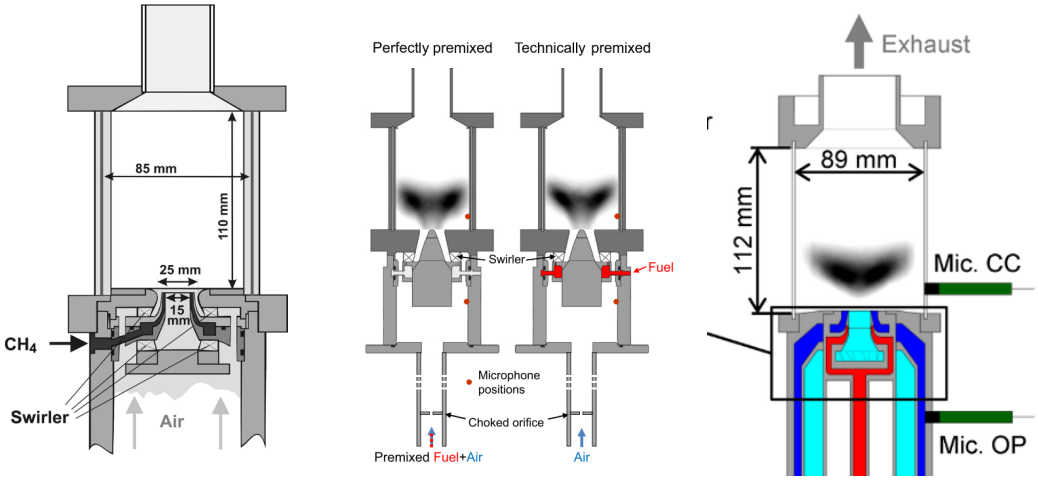


Figure 1.4: Various GTMC designs: Meier burner [3] (Left) PRECINSTA burner [4](Center) Arndt burner [5] (Right).

As these combustors became more widespread, there was an increased collaboration between these industrial-developed GTMCs and academic institutions. As these designs became more friendly regarding complexity and operating cost, academic institutions began to develop and study their examples of GTMC.

Gas turbine model combustors (GTMCs) have been developed to enhance understanding the underlying physics inherent to practical gas turbine systems in a laboratory environment and act as validation cases for developing modeling capabilities. These laboratory burners have spanned various geometries, injection schemes, and stabilization strategies. A comprehensive collection of GTMC experimental configurations can be found in Stohr et al. [15]. Of particular note is the family of burners studied at the German Aerospace Center (Deutsches Zentrum für Luft-und Raumfaht(DLR)). These have included the PRECCINSTA burner [4], the dual-swirl burner [3, 6, 16], and most recently the independent dual-swirl burner [5, 17, 18].

The primary burner examined is the DLR dual-swirl burner developed by Meier et al. [16, 3] This burner deviates from previous designs by virtue of having multiple swirlers and a partially-premixed combustion regime. Three distinct operating conditions characterize the dual swirl burner: A, B, and C. Flame A exhibited a stable V-shaped flame. In contrast, flame B operated as an unstable flat flame with peak instability at 280-300 Hz. Finally, flame C operated near the flammability limits with periodic blowout

and re-ignition observed. These conditions have been examined using stereoscopic particle image velocimetry (stereo-PIV), Raman spectroscopy, OH*/CH* chemiluminescence, and OH/CH/CH2O planar laser-induced fluorescence (PLIF). These measurements were used to characterize the burner's steady and unsteady performance. Studies included swirl number dependencies, flow structure development [19], precessing vortex core behavior, unsteady local mixing [20], and vortex flame interaction [15]. In addition to the original work conducted by DLR, an identical setup was investigated by Allison et al. [21] which focused on the behavior of the burner when operating at various equivalence ratios and particularly with more complex fuels, most notably syngas.

Various modeling efforts have been attempted on this geometry in concert with these experimental measurements. To the authors' knowledge, the first attempt was the work of Widenhorn et al. [22], which simulated the whole geometry under the flame A conditions and showed reasonable average velocity field comparison. A more comprehensive modeling effort was conducted by See and Ihme [23], who used a modified flamelet model and showed good agreement in the flame A averaged field condition. Additional work by Koo et al. [24] focused on accurately modeling soot formation in the system. These works primarily focused on the turbulent combustion modeling of the stable flame A configuration, focusing on flamelet model augmentation and particulate matter generation. Recent work by Chen et al. [25, 26] comprehensively modeled both the flame A and B conditions, focusing on the instability and validation of both the stable and unstable flame operating conditions representing the first complete steady and unsteady modeling of this system.

1.4 Computational Modeling

1.4.1 Large-Eddy Simulation (LES)

With recent advances in computational modeling, detailed simulations can be used to investigate the mechanisms of combustion instability for complex geometries. Especially, Large-Eddy Simulation (LES) have the potential to provide valuable insight into the

underlying unsteady dynamics. Huang et al. [27, 28] applied LES to a standard lean-premixed (LP) combustor [29]. Since these initial investigations, LES techniques have advanced as a modeling tool in combustion instability-prone systems. Within gas turbine-type systems, these simulation studies have been applied to a variety of problems ranging from laboratory combustors for both atmospheric [23] and high-pressure [30] conditions to (albeit under-resolved) studies of large-scale practical gas turbines [31]. The success of these methods in correctly characterizing combustion instability has varied significantly. The key modeling assumptions and problem complexity have had the greatest impact on modeling success. Regardless of accuracy, applying LES to these problems requires significant computational resources, ranging from 100,000 to 100 million CPU hours.

1.4.2 Reduced Modeling Using Physics-based Models

A major constraint in the design cycle analysis of combustion systems is turnaround time. As a result, many physics-based lower-order models have been used to reduce computational costs significantly.

Based on the wave or Euler equations analysis, the most common family of low-fidelity models are acoustic solvers [9]. These methods are popular due to their simplicity and fast solution time and have significant capability in purely acoustic flows. Under ideal conditions (simple geometries and mode shapes), the spatial and temporal modes can be separated. The spatial modes can be solved via the Helmholtz equation, and the temporal modes can be reduced to a set of ordinary differential equations. Using this, dominant eigenmodes can be determined together with growth rates. However, without the non-linear effects, these instabilities will be predicted to grow infinitely.

Regardless of the linear or non-linear analysis, a functional form relating heat addition and pressure perturbation is required. The most common prescribed function is that of the Crocco's $n-\tau$ model [32]. This method relates the heat release to pressure fluctuations via a time delay τ and an amplification factor n.

A data-driven flame transfer function (FTF) [33] method leverages experimental and simulation data to create the required relationship. While successful, this family of meth-

ods requires a transfer function to be computed due to the underlying assumption of a linear frequency domain. Thus, applications are generally restricted to combustion systems with low-amplitude perturbations, although successes have been noted in non-linear problems [34]. The flame describing function(FDF) [35, 36, 37] approach has shown success in extensions to large amplitude non-linear problems. However, it is recognized [38] that to compute an accurate FTF/FDF, a wide variety of input parameters are required. These may include parameters that are inaccessible or difficult to measure. Additionally, detailed non-linear dynamics significant to the underlying instability, like reactant mixing, cannot be accounted for. Ultimately the major concern with applying FTF/FDF methods is that they are difficult to extend to previously unknown problems.

1.4.3 Projection-based Reduced-Order Modeling (ROM)

Though LES is becoming affordable for complex problems and can reveal details of the underlying physics inaccessible through experiments, these techniques are still far out-of-reach for use in many-query applications (e.g., design, optimization, and uncertainty quantification). Therefore, it is imperative to develop reduced models that can inherit the fidelity of the LES, while being much more efficient for many-query computations. One class of these methods is the projection-based reduced-order model (ROM), which attempts to develop a dynamical system with reduced dimension to represent the full-order model (FOM) (e.g., LES). These methods have been proven effective in reducing the flow dynamics [39, 40, 41]. However, projection-based ROMs suffer in accuracy and stability when applied to multi-scale problems containing transport phenomena such as convection. These shortcomings can arise from a combination of the numerical stability of the projection (e.g., Galerkin [42]), the truncation of low-energy modes [43], and the inefficiency of linear manifolds in representing convective phenomena [44].

Several strategies have been suggested to mitigate these challenges. Balanced proper orthogonal decomposition has been used in linear systems to form stable ROMs [45, 46]. Attempts have also been made to improve ROM stability by examining the underlying numerical discretization through various methods. Rowley et al. [47] ensured that the

computed inner product for the Galerkin projection is physically meaningful. Parish et al. [48] used the Mori-Zwanzig formalism to develop a Petrov-Galerkin-type closure. Ahmed et al. [49] present a comprehensive review of state of the art in closure modeling for projection-based ROMs. It has been shown that maintaining the conservation properties of the governing equations is critical in ROM development [50]. A method demonstrated by Carlberg [51] generates stable non-linear ROMs by minimizing the least-squares residual of the projected solution, yielding a symmetrized and linearly stable ROM. This popular method is referred to as the Least-Squares Petrov Galerkin (LSPG) method.

In the context of reacting flow simulations, several additional challenges arise. A key challenge in computational combustion has been the numerical stiffness inherent in kinetics. Spurious oscillations have been observed near the high gradient conditions present in a flame front when modeled using ROM methods. These oscillations commonly lead to non-physical features such as negative temperature. Physical constraints have been applied to promote local stability with success. [52]

The model-form preserving least-squares projection with variable transformation (MP-LVST) [53] is a ROM formulation that allows conservation to be maintained while using alternate variables. This change of variables is beneficial in the reacting flow as using primitive variables (pressure, temperature, velocity) allows for simplified state calculations compared with conserved forms (density, enthalpy, momentum). This transformation is combined with least squares (to improve global stability), physical constraints (to improve local stability), and hyper-reduction methods (to achieve computational speed-up).

These developments have significantly increased the accuracy and robustness of this family of projection-based ROMs. However, all these techniques fundamentally operate by projecting a high-dimensional system onto a reduced manifold. As a result, the choice of reduced space is significant in ROM development.

The most common technique to develop the linear manifold is to use proper orthogonal decomposition [54] based on snapshots of the full-state vector. Generally speaking, problems with a strong limit cycle coherence have succeeded compared to chaotic problems

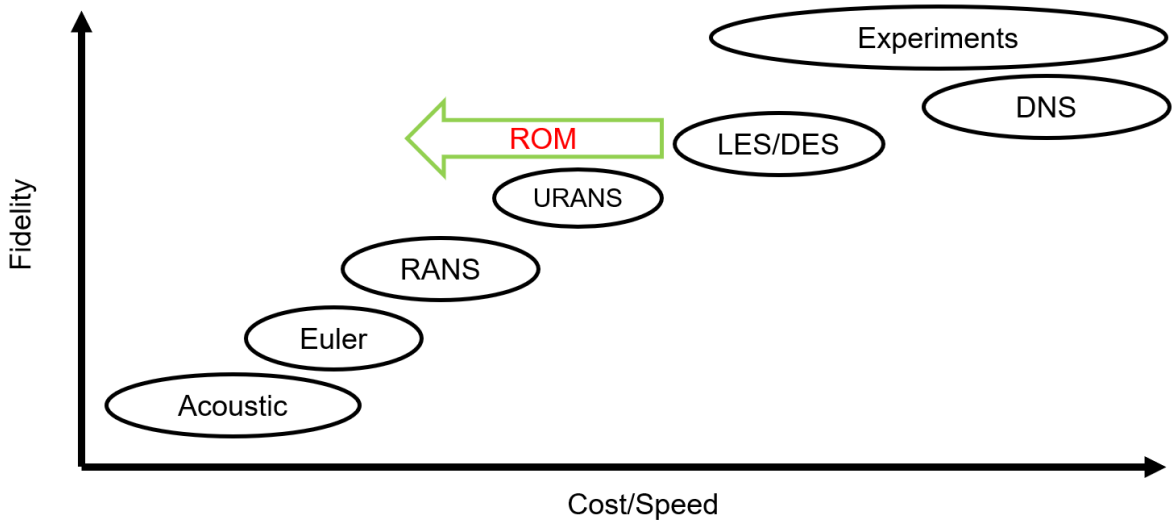


Figure 1.5: Heirarchy and trade-offs of various fluid flow computational methods.

with limited magnitude coherence [55].

1.5 Potential Use Cases for Projection ROMs

Various physics-based computational tools exist for fluid simulations with the general trend of providing more physical fidelity at the expense of computational resources (Fig. 1.5). The highest fidelity of simulation for complex geometry is most commonly that of Large-Eddy Simulation (LES). Exercising this level of fidelity involves significant man- and computer- hour requirements. Significant effort must be put into properly applying modeling and numerical techniques for a new problem type or application. Mesh generation on its own can be a month-long endeavor. All of these costs are even before the online computing phase of a simulation. Even at this point, during the online phase, assumptions or procedures may need to be revisited and adjusted as the simulation proceeds. Together these costs are, at minimum many months to years, depending on the resources invested. As a result, LES is primarily used for posterior analysis in cases of failure or for analysis of existing developed systems.

1.6 Current State-of-the Art and Grand Challenge

As a result of the experimental and computational efforts on reacting flows, there is a wealth of expertise in the understanding and modeling of combustors. However, current computational models are either highly simplified to a particular problem type (reduced fidelity) or require many months to conduct a high-fidelity simulation for a single operating and design point. Ultimately, engineers need access to high-fidelity responses to design changes with the computational wall time of reduced models. The inherent need for transient and operational analysis of these devices for combustion instability-prone problems compounds the required computations. In conjunction with the capabilities and limitations of models, one must consider the resources needed for high-fidelity simulations in terms of access to high-performance computing resources and the expertise and cost to operate them. Reduced-order modeling, on the other hand, has promised significant computational speed up (10-1000 times faster) but, until recently, has been limited to simple problems. In particular, very little work on ROMs has focused on the practical scientific computing required to apply these methods on a larger scale. With these in mind, the grand challenge of reduced order modeling is maintaining the fidelity and accuracy of full-fidelity simulations while reducing the query time and resource cost (both in the acquisition of compute resources and cost per compute hour). (Fig 1.5)

1.7 Thesis Goals and Contributions

1. **Development of Reduced-Order Modeling Techniques for Combustion Problems** With the current capabilities and limitations of the current state of the art, this thesis pursues model reduction via the development of scalable projection-based reduced-order models (ROMs) for combustion-type problems. Historically these methods have not been applied to fluid flows of significant complexity. This thesis develops and exhibits these methods on combustion-type problems and makes significant strides in three critical areas.

(a) **Efficiency:** Projection-based ROM methods provide a consistent framework for generating a reduced set of dynamical equations. However, dimension-reduction results in dense linear algebra and processing of the full-state operator. This fundamentally limits the computational efficiency of the reduced model and the computers needed to solve them. This thesis describes methods for overcoming this limitation using sparse-sampling techniques, and their performance is quantified.

(b) **Prediction:** For ROMs of this nature to be truly useful for engineering applications involving chaotic behavior, they must be capable of predicting flow scenarios that they were not initially trained on. This thesis shows that using adaptive- basis and sampling techniques significantly improves the capability of these ROMs to be genuinely predictive for complex and chaotic problems.

(c) **Scalability:** There has been a minimal description of the implementation methods needed to apply the ROM framework to problems of significant scale. Every aspect of this thesis is built upon high-performance computing technology and, in particular, the development of an overall software framework that enables the use of all the aforementioned techniques for complex and large problems.

2. **Application of ROM methods to large Gas Turbine Model Combustors (GTMC)** Modeling and understanding Gas Turbine Model Combustors (GTMCs) remains a significant challenge. Even high-fidelity large-eddy simulations (LES) are not guaranteed to be accurate. In this thesis, a GTMC setup will be explored from the perspective of a traditional CFD investigation. This GTMC is the well-documented DLR Dual-Swirl Burner developed by Meier [16]. LES simulations are performed with a flamelet progress variable approach and validated against available experimental data. Good agreement is shown in both steady and unsteady operations. Finally, the operational challenges of leveraging these simulations in an engineering setting will be described. From the perspective of addressing the grand challenge mentioned above, high-fidelity simulations form the basis for building

reduced-order models and performing high-dimensional simulations at a fraction of their current computational cost.

3. **Scalability of methods: PLATFORM and Load Balanced Adaptive Sampling** Significant challenges exist in applying these methods to larger problems.

   (a) **Pre-Processing:** Projection-based ROM requires pre-processing that scales aggressively with problem size to the order of 100s of Tera-bytes. The Parallel Linear Algebra Tool FOr Reduced Modeling (PLATFORM) was developed to facilitate this. This software was developed to assist to handle the parallel linear algebra and distributed I/O challenges associated with ROM development, both on and offline. In addition to being a core part of the ROM framework, distributed linear algebra enables advanced modal decomposition analysis and generic analysis of large datasets.

   (b) **ROM Solver Integration:** The second challenge is associated with the in-situ sampling adaptation for ROM methods. The sampling methods used to improve ROM efficiency yield diminishing returns due to poor load balancing. Addressing this represents a unique challenge as it requires in-situ redistribution of non-contiguous mesh elements. The methods and software methodology used to overcome this load imbalance and computational efficiency metrics are described and analyzed.

   Finally, recommendations are made for best practices and solver architectures that will allow significantly easier integration of ROM solvers. This will be discussed in the context of ROM-FOM coupling and computational frameworks and resources.

1.8 Thesis Outline

The remainder of the thesis is structured as follows:

- **Chapter 2**: Introduces projection-based reduced-order model (ROM). The fundamental derivations and equations are explained, emphasizing the application to

combustion problems. These derivations will span from classical Galerkin ROM to newer formulations that will be applied. The traditional computational combustion large-eddy simulation (LES) methods are described. The LES solver is used for data generation for the ROMs explored and the solver in which the ROM methods are implemented.

- **Chapter 3**: Explores the offline training requirements for the projection-based ROMs. PLATFORM, the software developed to handle these requirements, is described, and the distributed computing strategies used are explained.

- **Chapter 4**: Describes the DLR GTMC experimental setup. The LES simulation results are compared with experimental studies and previous computational studies. Finally, the reduced-order model results are compared with the computational results.

- **Chapter 5**: Introduces the adaptive sampling method. The described method is applied to two test cases, a 1D laminar flame, and 2D rocket injector. The efficiency and accuracy are quantified based on the ROM hyperparameters.

- **Chapter 6**:The load-balancing framework is introduced, and the computational method is described. The load-balanced method is compared to the same test problems and quantifies performance.

- **Chapter 7**: The results and conclusions of the previous sections are collated. These will include the feasibility of the application of these methods to problems of this type and scale and validation of the methods vs. experimental methods. Finally, the requirements for industrial application for these methods are discussed.

1.9 Notation

Consistent notation is used unless otherwise stated. Scalar and scalar returning functions are referred to by lowercase, italicized Latin or Greek letters (x, α). Vectors or corresponding functional operators are denoted using bold lowercase, italicized Latin or

Greek letters ($\mathbf{x}, \phi$). Matricies or corresponding functional operators are denoted using bold uppercase, italicized Latin or Greek letters ($\mathbf{X}, \mathbf{\Phi}$). Matrices and vector dimensions are noted via set definitions e.g., $\in \mathbb{R}^N$ is a real vector of dimension N. "=" is used to denote equality, while "$\triangleq$" is used to denote a definition. Parenthesis "()" are used to define function parameters, e.g., $f(x)$, where f is a scalar function operating on scalar x. The mathematical order of operation priority is dictated using brackets "[]". Superscripts x^2 are reserved for mathematical exponentiations. An exception is to note value perturbations which are noted using a single quotation p'.

Chapter 2

Computational Methods and Theoretical Background

2.1 Outline

In chapter 1, a variety of computational modeling techniques aimed at reducing the complexity of modeling reacting flow systems were discussed. These methods revolve around reducing the cost and time compared to the highest fidelity simulations. Within this dissertation, Large-Eddy Simulation (LES) is considered the highest fidelity of practical simulation for our systems of interest and is referred to as the Full-Order Model(FOM). It should be noted that most so-called reduced-order models described in the previous chapter do not leverage formal dimensional reduction but rather modeling assumptions or simplifications. Projection-based reduced-order models which leverage a mathematical order reduction are the primary focus of this thesis and, for simplicity, will be referred to simply as ROMs.

In this chapter, the reduced-order models developed in the thesis will be introduced. The general formulations existing in the literature as well as relevant strategies for reacting flow problems will be discussed. As will be seen later the execution of these types of ROMs requires the evaluation of more classical fluid modeling, in the form of LES. The computational methods of the LES code used will be detailed and explained. The focus of this chapter is on the computational equations with details of the equally critical implementation explored in later chapters.

2.2 Projection

The concept of order reduction is based on a low-dimensional representation of the larger full-order system. Consider a vector representing the state of a system.

$$\mathbf{q} \in \mathbb{R}^M \tag{2.1}$$

For a linear projection, we aim to represent the full dimensional state as a linear combination of a set of basis vectors $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_k] \in \mathbb{R}^{M \times k}$.

$$\mathbf{q}(t) \approx \tilde{\mathbf{q}}(t) \triangleq \sum_{i=1}^{k} \mathbf{v}_i \hat{\mathbf{q}}_i(t) \tag{2.2}$$

The reduced state $\hat{\mathbf{q}} \in \mathbb{R}^k$ is the resulting reduced or latent state vector (sometimes referred to as modal coefficients).

The core to the ROMs pursued in this thesis is predicated on the choice of trial basis $\mathbf{V}$ where $k << M$. This leads to a compressed representation of the state, reducing the dimension by orders of magnitude.

2.2.1 Suitability of Projection

To give a physical intuition about linear projection and its capabilities, consider a 1D domain with two wave types visualized in figure 2.1. The first is that of a standing wave. This feature will oscillate in magnitude however, spatially will remain anchored in place. As a result, if we have a mode given by that wave shape, we can describe any time instance of the standing wave via modulating the modal coefficient $\mathbf{a}$. In contrast, a traveling wave cannot be easily described using several linear modes. In its most naive form, we would require many modes that can be linearly combined to form the traveling wave front. This fundamental requirement is why linear projection-based ROMs have been relatively limited in their application to convective-type physics. In the context of combustion instability, using this method can have strengths as the quarter wave oscillation characteristic of combustion instability would be amenable to linear projection; however,

a large amount of convective hydrodynamic and acoustic properties would fundamentally limit the method.

2.3 Reduced-Order Modeling (ROM)

2.3.1 Projection-Based ROMs

Projection-based reduced-order models (ROMs) seek a low-dimensional representation of a dynamical system state and a projection of the governing equations onto the low-dimensional space to reduce the computational cost of numerical solutions for high-dimensional systems. The system state $\mathbf{q} \in \mathbb{R}^M$ can be represented as a linear combination of a set of orthogonal vectors $\mathbf{V} \in \mathbb{R}^{M \times k}$. This approximation takes the form

$$\mathbf{q} \approx \mathbf{V}\hat{\mathbf{q}} \tag{2.3}$$

In an ideal scenario, a combination of vectors can perfectly represent the full-dimension state ($\mathbf{q}$). Realistically the combination will lead to error from the dimension reduction.

$$\mathbf{q} = \mathbf{V}\hat{\mathbf{q}} + \mathbf{r} \tag{2.4}$$

This residual can be considered information lost via the order reduction and is referred to as the projection error. Ultimately the quality of the trial basis $\mathbf{V}$ sets an absolute upper limit in terms of accuracy for any methods leveraging this form of reduced-order modeling.

2.3.2 ROM formulation

We now describe how the general linear projection is used to create a ROM; we begin with a general non-linear dynamical system ODE,

$$\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} = \mathbf{f}(\mathbf{q}, t), \quad \mathbf{q}(t_0) = \mathbf{q}_0, \tag{2.5}$$

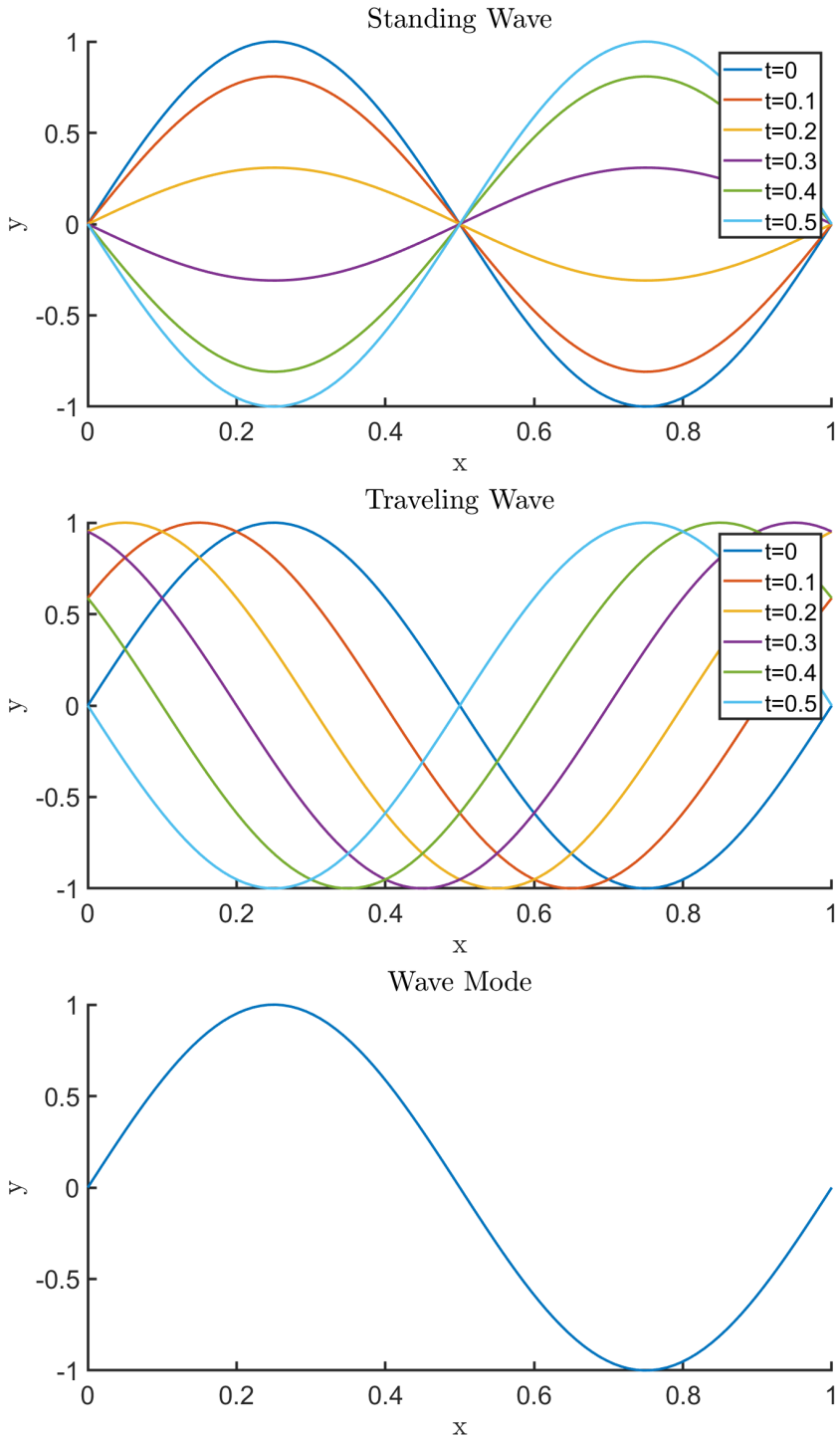


Figure 2.1: Standing Wave (top) Traveling Wave (Middle) Mode Shape (Bottom).

where $\mathbf{q} \in \mathbb{R}^M$ represents the system state, and $\mathbf{f}$ represents the non-linear function that describes the spatially discretized terms in the original governing PDE. For realistic systems, the dimension of this system M can be substantial ($N_{vars} \times N_{cells}$). For work within this thesis, we look at problems with a M of up to two hundred million. Repeatedly solving this system for different operating conditions is prohibitively expensive and requires extensive computing resources. The ultimate goal of model order reduction techniques is to reduce the dimension of equation 2.5 without significantly compromising the accuracy of the underlying dynamics.

Starting from equation 2.3 we introduce a modified approximation,

$$\mathbf{q}(t) \approx \tilde{\mathbf{q}} = \overline{\mathbf{q}} + \mathbf{P}^{-1}\mathbf{V}\hat{\mathbf{q}}(t), \tag{2.6}$$

where $\overline{\mathbf{q}} \in \mathbb{R}^M$ represents a constant reference state, $\mathbf{V} \in \mathbb{R}^{M \times k}$ is the *trial* basis composed of k linearly independent vectors, $\mathbf{P} \in \mathbb{R}^{M \times M}$ is a diagonal matrix representing the normalization of different variables in the state $\mathbf{q}$, and $\hat{\mathbf{q}} \in \mathbb{R}^k$ represent the modal coefficients describing the linear combination of the trial basis vectors describing the approximate state of $\tilde{\mathbf{q}} \in \mathbb{R}^M$. The normalization matrix $\mathbf{P}$ is formulated to make the state variables equivalent in contribution. Without normalization, variables such as pressure and energy may have very high magnitudes compared to species mass fractions and velocity.

The governing equation is first normalized and then projected onto the trial space as

$$\mathbf{V}^T\mathbf{P}\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} = \mathbf{V}^T\mathbf{P}\mathbf{f}(\mathbf{q}, t). \tag{2.7}$$

Recognizing that $\mathbf{P}$, $\overline{\mathbf{q}}$, and $\mathbf{V}$ are constant in time, and substituting Eq. 2.6, we are able to simplify this to

$$\mathbf{V}\frac{\mathrm{d}\hat{\mathbf{q}}}{\mathrm{d}t} = \mathbf{P}\mathbf{f}(\tilde{\mathbf{q}}, t) \tag{2.8}$$

Here we observe that $\mathbf{V}\frac{\mathrm{d}\hat{\mathbf{q}}}{\mathrm{d}t} \in \mathbb{R}^M$ and no dimension reduction has occurred. Finally, we project the system onto a low-dimensional *test space*, via the *test basis* $\mathbf{W} \in \mathbb{R}^{M \times k}$.

Similar to the trial basis $\mathbf{V}$, the test basis $\mathbf{W}$ is a set of k linearly independent vectors. The resulting system takes on the form which is considered the general ROM equation,

$$\mathbf{W}^T\mathbf{V}\frac{\mathrm{d}\hat{\mathbf{q}}}{\mathrm{d}t} = \mathbf{W}^T\mathbf{P}\mathbf{f}(\tilde{\mathbf{q}}, t), \tag{2.9}$$

where an ODE system of reduced dimension k has been achieved. This reduced system can be advanced using standard time-integration schemes (explicit or implicit).

The choice of the bases $\mathbf{W}$ and $\mathbf{V}$ are critical in the success of the ROM, and the selection of formulation of these bases has been the focus of a wide variety of ROM development. Typically the trial basis $\mathbf{V}$ is computed utilizing the proper orthogonal decomposition (POD) of the data-set trajectory. The computation of the trial basis and associated challenges are discussed in chapter 3. The choice of the test basis is therefore the critical formulation element of the developed ROM.

2.3.3 Galerkin ROM

In the situation where the trial basis equals the test basis $\mathbf{W} = \mathbf{V}$ Eq. 2.9 simplifies to

$$\frac{\mathrm{d}\hat{\mathbf{q}}}{\mathrm{d}t} = \mathbf{V}^T\mathbf{P}\mathbf{f}(\tilde{\mathbf{q}}, t) \tag{2.10}$$

which is referred to as a Galerkin projection.

2.3.4 Least-Squars Petrov-Galerkin

Another method seeks to formulate the test basis $\mathbf{W}$ so that the fully-discrete FOM residual $\mathbf{r}$ is minimized, which is referred to as the least-squares Petrov-Galerkin (LSPG) projection [51]. For a generic linear multi-step method with s steps the $\mathbf{r}$ is given as

$$\begin{aligned}\mathbf{r}^n = a_0\mathbf{q}^n + \sum_{i=1}^{s} a_i\mathbf{q}^{n-i} - \Delta t\beta_0\mathbf{f}(\mathbf{q}^n, t) \\ -\Delta t\sum_{i=1}^{s}\beta_i\mathbf{f}(\mathbf{q}^{n-i}, t^{n-i}),\end{aligned} \tag{2.11}$$

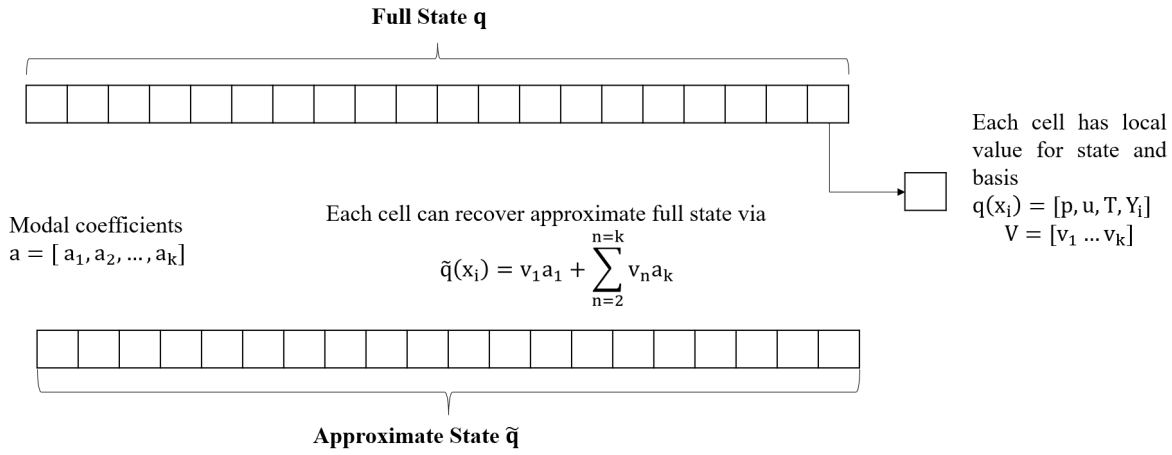


Figure 2.2: Schematic of approximate state evaluation from basis modes.

where a_i and β_i are dependent on the time integration scheme.

The minimization problem for solving the ROM at time instance n is posed as

$$\begin{aligned}\tilde{\mathbf{q}}^n &= \underset{\tilde{\mathbf{q}}^n \in \text{Range}(\mathbf{V})}{\text{argmin}} ||\mathbf{Pr}(\tilde{\mathbf{q}}^n)||_2^2 \\ &= \underset{\tilde{\mathbf{q}}^n \in \text{Range}(\mathbf{V})}{\text{argmin}} \|(\mathbf{Pr})^T(\mathbf{Pr})\|_2^2\end{aligned} \tag{2.12}$$

The solution to this problem is

$$(\mathbf{W}^n)^T \mathbf{Pr}(\tilde{\mathbf{q}}^n) = 0, \tag{2.13}$$

where

$$\mathbf{W}^n = \frac{\partial \mathbf{Pr}(\tilde{\mathbf{q}}^n)}{\partial \hat{\mathbf{q}}^n}. \tag{2.14}$$

For a linear multi-step method (Eq. 2.11) this results in a test basis of the form

$$\mathbf{W}^n = \mathbf{P}[\mathbf{I} - \Delta t \beta_0 \frac{\partial \mathbf{r}^n}{\partial \tilde{\mathbf{q}}^n}]\mathbf{P}^{-1}\mathbf{V}. \tag{2.15}$$

Notably, for an explicit integrator $\mathbf{W}^n = \mathbf{V}$ reduces to the Galerkin projection. As a result of the symmetrization provided by this formulation, LSPG has shown significant improvement over the classical Galerkin projection for convective-type problems. This method has been successfully used for a variety of non-reacting flow applications.

Both of these methods face challenges [55] in their application to reacting flow problems. In particular, both methods have been shown to exhibit non-physical oscillations and generally suffer from stability problems associated with the stiffness of chemical kinetics and the associated high gradients [55]. Further details on the stability characteristics of Galerkin and LSPG methods for problems of this type can be found in Huang et al. [53].

2.3.5 Model-Form Preserving Least-Squares with Variable Transformation

A more recent variant of the LSPG method of ROM formulations is the model-form preserving least-squares variable transformation(MP-LSVT) [53]. The MP-LSVT formulation uses a combination of approaches to improve the robustness of the ROM. The variable transformation allows for symmetrization (and hence global linear stability) while maintaining discrete consistency and structure preservation. Local stability is enhanced using physical limiters. MP-LSVT allows one to represent the system state as a function of primitive variables $\mathbf{q}_p$. An additional normalization matrix $\mathbf{H}$ is applied to the primitive variables to improve the conditioning of the system further as

$$\tilde{\mathbf{q}}_p(t) = \overline{\mathbf{q}}_p + \mathbf{H}^{-1}\mathbf{V}_p\hat{\mathbf{q}}_p(t). \tag{2.16}$$

We then define a state transformation function $\mathbf{q} : \mathbf{q}_p \mapsto \mathbf{q}$, which maps the primitive variables to their corresponding conservative form. This updates Eq. 2.5 as

$$\frac{\mathrm{d}\mathbf{q}(\mathbf{q}_p)}{\mathrm{d}t} = \mathbf{f}(\mathbf{q}, t). \tag{2.17}$$

Here we note that the evaluation of the non-linear function uses a combination of conservative and primitive variables but is noted just as $\mathbf{q}$ for convenience. This leads to a corresponding form of the residual as

$$\mathbf{r}(\mathbf{q}_p^n) = a_0\mathbf{q}(\mathbf{q}_p^n) + \sum_{i=1}^{s} a_i\mathbf{q}(\mathbf{q}_p^{n-i})$$
$$-\Delta t\beta_0\mathbf{f}(\mathbf{q}^n, t) - \Delta t\sum_{i=1}^{s}\beta_i\mathbf{f}(\mathbf{q}^{n-i}, t^{n-i}). \tag{2.18}$$

when this updated residual is applied to the minimization problem from the LSPG description,

$$\mathbf{W}_p^n = \frac{\partial \mathbf{P}\mathbf{r}(\tilde{\mathbf{q}}_p^n)}{\partial \hat{\mathbf{q}}_p^n}, \tag{2.19}$$

yields the test basis,

$$\mathbf{W}_p^n = \mathbf{P}[\mathbf{\Gamma}^n - \Delta t\beta_0\mathbf{J}]\mathbf{H}\mathbf{V}_p \tag{2.20}$$

where $\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{q}_p}$, and $\mathbf{\Gamma} = \frac{\partial \mathbf{q}}{\partial \mathbf{q}_p}$.

2.3.6 Hyper-Reduction/Sampling

While the above methods have shown recent success in improving the robustness of ROM methods for combustion-driven flows, little computational savings are achieved purely from the projection. Examining equation 2.9, we see the non-linear function $\mathbf{f}$ must still be evaluated at the full dimension. A well-studied hyper-reduction method for this family of ROMs is the discrete empirical interpolation method (DEIM). DEIM, introduced by Chaturantabut and Sorensen [56], introduces an approximate form of the FOM non-linear ($\mathbf{f}$) function as

$$\mathbf{f}(\tilde{\mathbf{q}}) \approx \mathbf{U}(\mathbf{S}^T\mathbf{U})^{-1}\mathbf{S}^T\mathbf{f}, \tag{2.21}$$

where $\mathbf{S} \in \mathbb{R}^{M\times s}$ is composed of s unique unit vectors. This sampling matrix selects a subset of points to evaluate full non-linear function. $\mathbf{U} \in \mathbb{R}^{M\times d}$ is an orthonormal basis. Ideally, this will be generated by a POD decomposition of snapshots of the FOM non-linear function $\mathbf{f}$. However, in our cases we use the trial basis $\mathbf{V}$. The trial basis has been shown to produce a good approximation and significantly simplify pre-processing.

Gappy-POD [57] takes a very similar form but with the relaxation of the requirement on the number of sampled points. The formulation then takes on a similar form with the introduction of the Moore-Penrose inverse. Applying this to the residual of the FOM

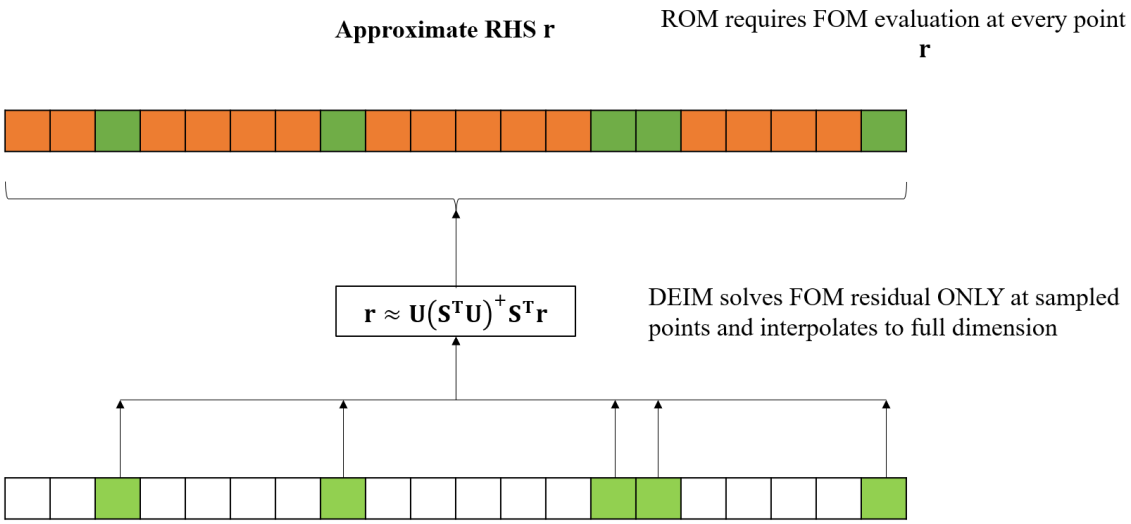


Figure 2.3: DEIM solves residual at sampled points $\mathbf{S}$ and interpolates the unsampled points.

solution (Eqn. 2.11) we get the form

$$\mathbf{f}(\tilde{\mathbf{q}}) \approx \mathbf{U}(\mathbf{S}^T\mathbf{U})^+\mathbf{S}^T\mathbf{f}, \tag{2.22}$$

which describes an approximate form of the full residual from a set of sampled points defined by $\mathbf{S}$. Applying the sparse reconstruction to the non-linear equation residual we recover

$$\tilde{\mathbf{q}_p^n} = \underset{\tilde{\mathbf{q}_p^n} \in \text{Range}(\mathbf{V}_p)}{\text{argmin}} ||\mathbf{U}(\mathbf{S}^T\mathbf{U})^+\mathbf{S}^T\mathbf{P}\mathbf{f}(\tilde{\mathbf{q}_p^n})||_2^2 \tag{2.23}$$

Which results in the test basis,

$$\mathbf{W}_p^n = \frac{\partial \mathbf{U}[\mathbf{S}^T\mathbf{U}]^+\mathbf{S}^T\mathbf{P}\mathbf{f}(\tilde{\mathbf{q}_p^n})}{\partial \mathbf{q}_r^n} = \mathbf{U}[\mathbf{S}^T\mathbf{U}]^+\mathbf{S}^T\frac{\partial \mathbf{P}\mathbf{f}(\tilde{\mathbf{q}_p^n})}{\partial \mathbf{q}_r^n} = \mathbf{U}[\mathbf{S}^T\mathbf{U}]^+\mathbf{S}^T\mathbf{W}_p^n \tag{2.24}$$

Ultimately this form means that instead of evaluating the full dimension M only a subset of points $\mathbf{f}$ needs to be calculated.

$$[\mathbf{S}^T\mathbf{W}_p^n]^T\left[[\mathbf{S}^T\mathbf{U}]^+\right]^T[\mathbf{S}\mathbf{U}]^+\mathbf{S}^T\mathbf{P}\mathbf{f}(\tilde{\mathbf{q}_p^n}) = 0 \tag{2.25}$$

This is the source of the majority of expected computational savings from projection-based ROMs and is visualized in figure 2.3.

Initial Sparse Sampling

The initial point selection $\mathbf{S}$ is computed using a GappyPOD plus eigenvalue method. [58] This form of GappyPOD was shown to perform the best for problems of this type. [52] The method first selects d elements using a rank revealing QR decomposition of the basis modes $\mathbf{U}$. However, because for a majority of ROMs the goal is the number of modes is relatively small, this only finds the first k of s desired sampled cells.

The remaining cells are found via an eigenvalue method. Examining the L2 norm of the sampling error

$$||[\mathbf{S}^T\mathbf{U}]^+||_2 = \sigma_{max}([\mathbf{S}^T\mathbf{U}]^+) = \frac{1}{\sigma_{min}(\mathbf{S}^T\mathbf{U})} \tag{2.26}$$

We aim to leverage the method to minimize the sampling error by consecutively choosing a row of $\mathbf{U}$ (or a physical degree of freedom) which maximizes the smallest singular value (or equivalent eigenvalue) of $\mathbf{S}^T\mathbf{U}$.

Later in the thesis, we will discuss in-situ resampling. This resampling means that the initial point selection is not as critical and thus the initial point selection is chosen randomly. This significantly eases pre-processing of larger size cases.

Finally, we note that since multiple degrees of freedom (DOF) are located at each physical cell, once a point has been selected, it cannot be selected again. That is to say, a call selected by its DOF will include all DOFs of its cell.

2.4 General ROM framework and terminology

Three major components can be thought of as the components that most strongly affect a developed ROM from a formulation perspective. These are the,

1. **Choice of test basis:** The choice of test basis $\mathbf{W}$ can be set as either Galerkin ($\mathbf{W} = \mathbf{V}$) or Petrov-Galerkin ($\mathbf{W} \neq \mathbf{V}$).

2. **Quality of trial basis projection:** The trial basis is generated from training data. The quality of this training region thus has the greatest impact on this aspect of

any developed ROM.

3. Type of Sampling Strategy: A variety of methods exist to choose the sampling points for hyper reduction. These various methods, especially for statically sampled ROMs can greatly impact ROM accuracy.

2.5 Combustion Modeling

The results presented in this thesis are obtained using the General Equations and Mesh Solver (GEMS) [59]. GEMS is an LES code that has been modified extensively for research studies of reduced-order models. This section will review the governing equations of the methods and note different representations of the equations. This is especially important in the context of ROM methods, as the contrast in dense and sparse linear-algebra techniques means that a less intuitive representation must be used to maintain consistency in some cases.

The continuity, conservation of momentum, and total enthalpy equations take the form,

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_j}{\partial x_j} = 0, \tag{2.27}$$

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_j u_i}{\partial x_j} + \frac{\partial p}{\partial x_i} - \frac{\partial \tau_{ij}}{\partial x_j} = 0, \tag{2.28}$$

and

$$\frac{\partial (\rho h^0 - p)}{\partial t} + \frac{\partial \rho u_j h^0}{\partial x_j} - \frac{\partial u_i \tau_{ij}}{\partial x_j} - \frac{\partial}{\partial x_j}\Big(\lambda \frac{\partial T}{\partial x_j}\Big) - \frac{\partial}{\partial x_j}\left(\rho \sum_{l=1}^{N} \left[\langle h_l \rangle D_{lM} \frac{\partial Y_l}{\partial x_j}\right]\right) = 0, \tag{2.29}$$

respectively. Here ρ, u, p, T, and h^0 represent the density, velocity, pressure, temperature, and total enthalpy respectively, λ is the mixture thermal conductivity; D_{lM} is the diffusion coefficient of species l into the mixture M. τ_{ij} is the resolved viscous stress

tensor assuming Newtonian fluid takes the form

$$\tau_{ij} = 2\mu\epsilon_{ij}, \tag{2.30}$$

where

$$\epsilon_{ij} = \frac{1}{2}\left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right] - \frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}, \tag{2.31}$$

with the bulk viscosity neglected.

Finally, in addition to these equations, any number of generic transport equations can be coupled with this generalized form

$$\frac{\partial \rho Y_i u_i}{\partial t} + \frac{\partial \rho u_j Y_i}{\partial x_j} - \frac{\partial}{\partial x_j}\left[\rho D \frac{\partial Y_i}{\partial x_j}\right] = \dot{\omega}_{Y_i}, \tag{2.32}$$

Here we represent the scalar as a mass fraction Y_i of some chemical species with a source term of $\dot{\omega}_{Y_i}$ representing the production or destruction of that species.

2.5.1 Vector Form

While the above notation is convenient for interpreting a given conservation variable in a coupled computational solver context, it can be helpful to represent the system in a coupled fashion instead of the state vector, flux, and source terms.

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot [\mathbf{f} - \mathbf{f}_{\text{visc}}] = \mathbf{h} \tag{2.33}$$

$$\mathbf{q} = \begin{bmatrix} \rho \\ \rho u_x \\ \rho u_y \\ \rho u_z \\ \rho h_0 \end{bmatrix} \tag{2.34}$$

For completeness, these inviscid ($\mathbf{f}$) and viscous ($\mathbf{f}_{\text{visc}}$) fluxes are given in as components in the x, y, and z directions.

$$\mathbf{f} = \mathbf{f}_x + \mathbf{f}_y + \mathbf{f}_z \tag{2.35}$$

$$\mathbf{f}_{\text{visc}} = \mathbf{f}_{\text{visc}x} + \mathbf{f}_{\text{visc}y} + \mathbf{f}_{\text{visc}z} \tag{2.36}$$

$$\mathbf{f}_x = \begin{bmatrix} \rho u_x \\ \rho u_x^2 + p \\ \rho u_x u_y \\ \rho u_x u_z \\ \rho u_x h_0 \\ \rho u_x Y_i \end{bmatrix} \mathbf{f}_y = \begin{bmatrix} \rho u_y \\ \rho u_x u_y \\ \rho u_y^2 + p \\ \rho u_y u_z \\ \rho u_y h_0 \\ \rho u_y Y_i \end{bmatrix} \mathbf{f}_z = \begin{bmatrix} \rho u_z \\ \rho u_x u_z \\ \rho u_y u_z \\ \rho u_z^2 + p \\ \rho u_z h_0 \\ \rho u_z Y_i \end{bmatrix} \tag{2.37}$$

$$\mathbf{f}_{\text{visc}x} = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{yx} \\ \tau_{zx} \\ u_x\tau_{xx} + u_y\tau_{yx} + u_z\tau_{zx} - q_x \\ -\rho D_{v,z} Y_i \end{bmatrix}$$

$$\mathbf{f}_{\text{visc}y} = \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{zy} \\ u_x\tau_{xy} + u_y\tau_{yy} + u_z\tau_{zy} - q_y \\ -\rho D_{v,y} Y_i \end{bmatrix} \tag{2.38}$$

$$\mathbf{f}_{\text{visc}z} = \begin{bmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ u_x\tau_{xz} + u_y\tau_{yz} + u_z\tau_{zz} - q_z \\ -\rho D_{v,z} Y_i \end{bmatrix}$$

where D_v is the effective velocity of the transport scalar Y due to diffusion.

The source vector $\mathbf{h}$ is given by

$$\mathbf{h} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \dot{\omega}_{Y_i} \end{bmatrix} \tag{2.39}$$

2.5.2 Equations of State

For cases where reacting flow scalars are not present, the Navier-Stokes, continuity, and energy equations are solved together with the equation of state.

$$p = p(\rho, T) \tag{2.40}$$

this equation of state is close with the ideal gas assumption equation of state

$$p(\rho, T) = \rho RT \tag{2.41}$$

Here R is the mixture gas constant given by $R = \frac{R_U}{MW}$, where MW is the gas's molecular weight and R_U is the universal gas constant.

2.5.3 Thermodynamic Properties

The thermodynamic state quantities of a given mixture computed via NASA polynomials tabulated by McBride [60]. For a given species, the enthalpy can be given by

$$\frac{h_i}{RT} = -\frac{a_1}{T^2} + \frac{a_2}{T} lnT + a_3 + \frac{a_4}{2}T + \frac{a_5}{3}T^2 + \frac{a_6}{4}T^3 + \frac{a_7}{5}T^4 + \frac{a_8}{T} \tag{2.42}$$

For a given mixture of n component mass fractions (Y_i) the mixture thermodynamic quantity can be defined as weighted sums:

$$h_{mixture} = \sum_{i=1}^{n} Y_i h_i \tag{2.43}$$

The total enthalpy used in the energy equation is then

$$h^0 = h + \frac{1}{2}(u_j u_j). \tag{2.44}$$

2.5.4 Empirical Transport Properties

In addition to the thermodynamic properties, the molecular transport properties are required to compute the thermal conductivity λ, viscosity μ, and species diffusivity D. The viscosity and thermal conductivity are computed as weighted sums of the constituent species in a manner similar to that of enthalpy.

$$\ln\mu = A\ln T + \frac{B}{T} + \frac{C}{T^2} + D \tag{2.45}$$

$$\ln\lambda = A\ln T + \frac{B}{T} + \frac{C}{T^2} + D \tag{2.46}$$

The constants (A, B, C, D) are tabulated for each species.

The mass diffusivity D of an individual species i into the mixture M is given by Curtiss and Hirshfelder [61]

$$D_i, M = \frac{1 - x_i}{\sum_{m \neq i} \frac{X_m}{D_{i,m}}} \tag{2.47}$$

this combines the binary diffusion coefficients between the constituent species as modeled by the Chapman-Enskog theory [62]. The binary diffusion coefficient of species i into species m is given by

$$D_{i,m} = \frac{0.0266}{p\left[\frac{\sigma_i + \sigma_m}{2}\right]^2 \Omega} \sqrt{T^3 \left[\frac{1}{M_i} + \frac{1}{M_m}\right]} \tag{2.48}$$

where the collision diameters(σ) of species are given in Angstroms. The diffusion collision integral Ω comes from Nuefeld et al. [63] as

$$\Omega \triangleq \frac{d_1}{\exp(d_2 T^*)} + \frac{d_3}{\exp(d_4 T^*)} + \frac{d_5}{\exp(d_6 T^*)} + \frac{d_7}{\exp(d_8 T^*)} \tag{2.49}$$

with empirical constants $\begin{bmatrix} d_1 & \dots & d_8 \end{bmatrix}$. The reduced temperature T^* is computed as

$$T^* = T\left[\frac{k_B}{\sqrt{\epsilon_i \epsilon_m}}\right] \tag{2.50}$$

where k_B is the Boltzmann constant and ϵ_i is a tabulate Lennard-Jones species energy.

2.6 Combustion Modeling

A majority of the work in this thesis are focused on the effects of reacting flow in both full- and reduced-order modeling. To achieve this two models are leveraged, finite rate (FR) chemistry and the flamelet progress variable approach (FPVA).

2.6.1 Laminar Finite Rate Chemistry

The finite rate chemistry model introduces additional transport equations for the chemical species needed for a defined chemical mechanism. These additional transport equations take the form,

$$\frac{\partial \rho Y_i}{\partial t} + \frac{\partial \rho u_j Y_i}{\partial x_j} = \nabla \cdot [\rho D \nabla Y_i] + \dot{\omega}_i. \tag{2.51}$$

Here $\dot{\omega}$ is the production or destruction of species i, and D is the diffusivity of species i into the mixture of the n total species.

The source terms ω of each of the species are calculated using a set of elementary chemical reactions. The set is referred to as a mechanism. For a global set of N species, they can be represented generally as a set of L reactions,

$$\sum_{i=1}^{N} \nu'_{i,j} X_i \Longleftrightarrow \sum_{i=1}^{N} \nu''_{j,i} X_j \quad \text{for} \quad j = 1, ..., L \tag{2.52}$$

where X_i is the ith species mole fraction and $\nu'_{i,j}$ and $\nu''_{i,j}$ represent the stoichiometric coefficients of the i-th species in the j-th reaction.

The net production rate $\dot{\omega}_i$ as seen in equation 2.51 is then given by

$$\dot{\omega}_i = \sum_{i=1}^{L} \left[[\nu'_{i,j} - \nu''_{i,j}][k_f, j \prod_{j=1}^{N} [X_i]^{\nu'_{i,j}} - k_b, j \prod_{j=1}^{N} [X_i]^{\nu''_{i,j}}] \right] \tag{2.53}$$

where $[X_i]$ is the concentration of species X_i and k_f and k_b are the Arrhenius rates of the species consumption/production,

$$k_f = AT^b e^{\frac{-E_a}{RT}}, \tag{2.54}$$

where A is the rate coefficient, b is the temperature dependency exponent and E_a is the activation energy for the reaction.

The corresponding backward reaction rate can be computed using the reaction equilibrium relation

$$K = \frac{k_f}{f_b} \tag{2.55}$$

The global set of reactions and the corresponding reaction rate constants are generally empirically generated sets from experiments and the different mechanisms used will be described together with the problem setups.

2.7 Turbulence Modeling

Favre-averaging $\widetilde{\phi(\vec{x},t)}$ is used for velocity u, enthalpy h, total enthalpy h^0, temperature T, and the flamelet transport scalars Z_m, Z_v, C defined as

$$\widetilde{\phi(\vec{x},t)} = \frac{\langle \rho(\vec{x},t)\phi(\vec{x},t)\rangle}{\langle \rho(\vec{x},t)\rangle} \tag{2.56}$$

where the mean quantities $\langle\phi\rangle$ are defined over a time interval Δt by ,

$$\langle \phi(\vec{x},t)\rangle = \frac{1}{\Delta t}\int_t^{t+\Delta t} \phi(\vec{x},t')dt'. \tag{2.57}$$

The continuity and conservation of momentum and total enthalpy equations take the form of

$$\frac{\partial\langle\rho\rangle}{\partial t} + \frac{\partial\langle\rho\rangle\widetilde{u}_j}{\partial x_j} = 0. \tag{2.58}$$

$$\frac{\partial\langle\rho\rangle\widetilde{u}_i}{\partial t} + \frac{\partial\langle\rho\rangle\widetilde{u}_j\widetilde{u}_i}{\partial x_j} + \frac{\partial\langle p\rangle}{\partial x_i} - \frac{\partial\left[\widetilde{\tau_{ij}} - \widetilde{u_i''u_j''}\right]}{\partial x_j} = 0, \tag{2.59}$$

$$\frac{\partial(\langle\rho\rangle\widetilde{h^0} - \langle p\rangle)}{\partial t} + \frac{\partial\langle\rho\rangle\widetilde{u}_j\widetilde{h^0}}{\partial x_j} - \frac{\partial\widetilde{u}_i\widetilde{\tau_{ij}}}{\partial x_j} - \frac{\partial}{\partial x_j}\Big(\lambda\frac{\partial\widetilde{T}}{\partial x_j}\Big) - \frac{\partial}{\partial x_j}\left(\langle\rho\rangle\sum_{l=1}^{N}\left[\langle h_l\rangle\left[D_{lM}\frac{\partial\widetilde{Y}_l}{\partial x_j} - V_i^c\widetilde{Y}_l\right]\right]\right) + \frac{\partial\widetilde{u}_i\widetilde{u_i''u_j''}}{\partial x_j} - \frac{\partial\langle\rho\rangle\widetilde{u_j''e''}}{\partial x_j} = 0, \tag{2.60}$$

respectively. Here V_i^c is the correction velocity for diffusion; λ is the thermal conductivity; D_{lM} is the diffusion coefficient of species l into the mixture M. In most flamelet-based model implementations these thermal transport quantities are calculated online based on the species composition provided by the flamelet table. However, the GEMS implementation recalculates these based on the current state and composition. $\widetilde{\tau}_{ij}$ is the resolved viscous stress tensor assuming Newtonian fluid takes the form

$$\widetilde{\tau_{ij}} = 2\mu\widetilde{\epsilon}_{ij}, \tag{2.61}$$

where

$$\widetilde{\epsilon}_{ij} = \frac{1}{2}\left[\frac{\partial\widetilde{u}_i}{\partial x_j} + \frac{\partial\widetilde{u}_j}{\partial x_i}\right] - \frac{1}{3}\frac{\partial\widetilde{u}_k}{\partial\widetilde{x}_k}\delta_{ij}, \tag{2.62}$$

with the bulk viscosity neglected. $\widetilde{u_i''u_j''}$ is the large-eddy simulation(LES) mean stress modeled using the Nicoud Sigma model [64],

$$\widetilde{u_i''u_j''} = \tau_{ij}^{SGS} - \frac{1}{3}\tau_{kk}^{SGS}\delta_{ij} = 2\langle\rho\rangle\nu_t\widetilde{\epsilon}_{ij}, \tag{2.63}$$

where the Leonard and cross-terms are neglected [65]. The enthalpy equation unresolved term is closed using the gradient-diffusion model using the turbulent viscosity and a turbulent Prandtl number Pr_t approximated as .7;

$$\widetilde{u_j''e''} = \frac{\nu_t}{Pr_t}\frac{\partial\widetilde{h}}{\partial x_j} \tag{2.64}$$

2.7.1 Sigma Turbulence Model

The Sigma turbulence model [64] models the sub-grid scale turbulent viscosity as

$$\nu_t = (C_\sigma \Delta)^2 \mathcal{D}_\sigma(\widetilde{u}), \tag{2.65}$$

where $\mathcal{D}_\sigma(\widetilde{u}) = \frac{\sigma_3(\sigma_1-\sigma_2)(\sigma_2-\sigma_3)}{\sigma_1^2}$. Here σ_i are the singular values of the resolved velocity gradient ordered greatest to least and Δ is the local cell size. The differential operator defines the subgrid-scale viscosity based on the singular values of the velocity gradient tensor ($\widetilde{g}_{ij} = \frac{\partial \widetilde{u}_i}{\partial \widetilde{x}_j}$). This model has been applied to other stabilized flame cases [66, 67] and has shown favorable comparison for the relatively small computation cost compared to transport equation-based methods.

2.7.2 Non-Premixed Flamelet Modeling

In addition to the finite rate model, for cases more affected by turbulent combustion effects, the flamelet progress variable approach [68](FPVA) model was utilized. The FPVA model is derived from the steady laminar flamelet model. This model treats a turbulent flame as an ensemble of laminar flames, assuming that the temporal scales of the reaction are significantly faster and separate from the convective diffusive processes. Under this assumption, turbulent effects only act to deform the flame front. While this may seem an improper assumption, it has been shown that the non-premixed flamelet is valid below Karlovitz numbers (the ratio of chemical to turbulent time scales) of 190. [69] This upper bound makes the steady flamelet model quite applicable to a variety of combustion system, particularly in the low-Mach regime.

Considering the flame front as a deformed one-dimensional flame as an alternate coordinate system along the mixture fraction space. This transformation leads to the unsteady flamelet equation,

$$\rho \frac{\partial Y_i}{\partial t} - \frac{1}{2}\rho \chi_Z \frac{\partial^2 Y_i}{\partial Z^2} = \dot{m} \tag{2.66}$$

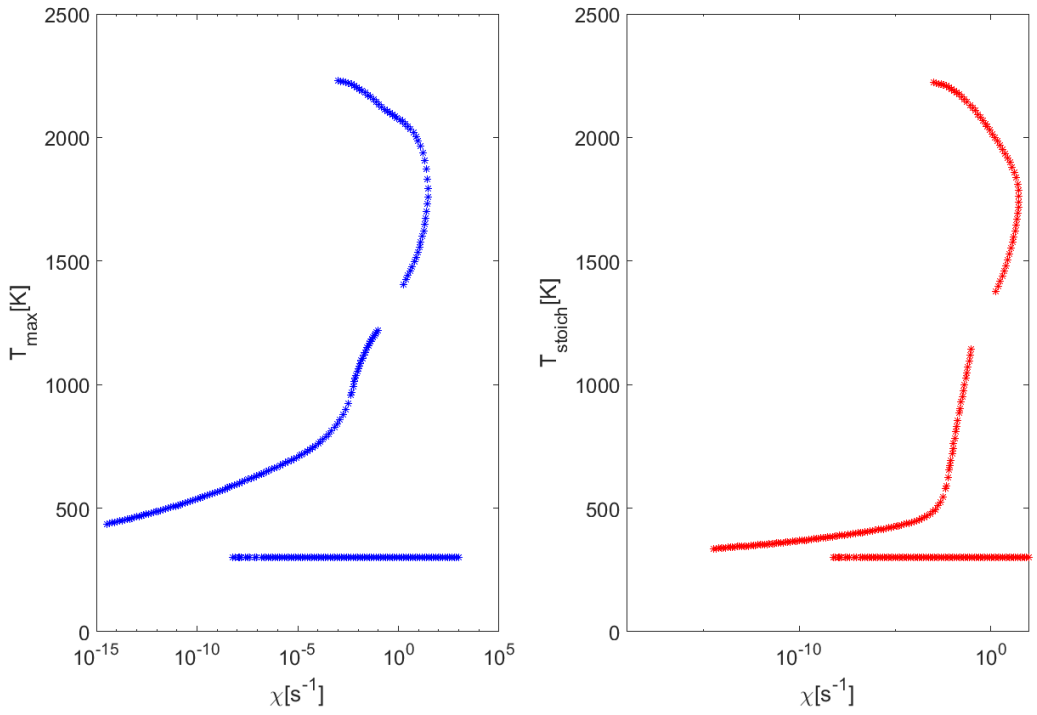


Figure 2.4: The S-shaped curve for GRI-1.2 methane-air kinetics with $T_{\text{fuel}} = 300$ and $T_{\text{ox}} = 300$. Each point represents a single steady flamelet solution.

where χ_Z is the scalar dissipation rate is given by $\chi_Z = 2D_Z|\nabla Z|^2$. A set of solutions to the steady form of this equation can then be computed for various values of scalar dissipation for a given chemical kinetic set. The different solutions of this equation will form the "S-shaped curve" parameterized by the system's scalar dissipation and stoichiometric temperature. An example of the curve for room temperature methane-air combustion is given in figure 2.4

Examining these solutions, we can observe that there are potentially up to three reasonable solutions for a given scalar dissipation. These correspond to stable burning (upper branch), unstable burning (middle branch), and a non-reacting state (lower branch). A mapping from the mixture fraction to the state properties can be proposed using this S-curve. For the original steady laminar flamelet model [70], only the upper branch was used with a direct parameterization of the scalar dissipation. However, this assumption using the stable burning branch of the "S-curve" is inadequate for partially premixed flames. The FPVA model instead adds parameterization of the progress variable (C), allowing access to all three branches of the S-curve. The progress variable can be specified

in various ways and indicates how complete a reaction is. The progress variable should exhibit monotonicity as the reaction progresses. Thus the most common definition is that of $C = Y_{CO_2} + Y_{H_2O} + Y_{CO} + Y_{H_2}$ which is the form used through out this thesis.

The individual flamelets were generated using FlameMaster [71] using the GRI-12 mechanism [72]. The flamelets were organized into a table for online computation with an adaptive mesh in the reaction zone.

The flamelet tabulation all occurs offline. In the context of the governing equations, the FPVA model reduces the n species equations into three scalar equations representing the Favre-filtered mean mixture fraction Z_m, mixture fraction variance Z''^2, and progress variable C.

$$\frac{\partial\langle\rho\rangle\widetilde{Z_m}}{\partial t} + \frac{\partial\langle\rho\rangle\widetilde{u}_j\widetilde{Z_m}}{\partial x_j} - \frac{\partial}{\partial x_j}\langle\rho\rangle\big(D + \frac{\nu_t}{Sc_t}\big)\frac{\partial\widetilde{Z_m}}{\partial x_j} = 0 \tag{2.67}$$

$$\frac{\partial\langle\rho\rangle\widetilde{Z''^2}}{\partial t} + \frac{\partial\langle\rho\rangle\widetilde{u}_j\widetilde{Z''^2}}{\partial x_j} - \frac{\partial}{\partial x_j}\left(\langle\rho\rangle\left[D + \frac{\nu_t}{Sc_t}\right]\frac{\partial\widetilde{Z''^2}}{\partial x_j}\right) = 2\langle\rho\rangle\Big[\underbrace{\frac{\nu_t}{Sc_t}\frac{\partial\widetilde{Z_m}}{\partial x_j}\frac{\partial\widetilde{Z_m}}{\partial x_j}}_{Production} - \underbrace{C_Z\frac{\nu_t}{\Delta^2}\widetilde{Z''^2}}_{Dissipation}\Big] \tag{2.68}$$

$$\frac{\partial\langle\rho\rangle\widetilde{C}}{\partial t} + \frac{\partial\langle\rho\rangle\widetilde{u}_j\widetilde{C}}{\partial x_j} - \frac{\partial}{\partial x_j}\left(\langle\rho\rangle\left[D + \frac{\nu_t}{Sc_t}\right]\frac{\partial\widetilde{C}}{\partial x_j}\right) = \langle\rho\rangle\widetilde{\dot{\omega}}_C \tag{2.69}$$

where Sc_t is the turbulent Schmidt number, approximated as 0.7. Traditional flamelet-based methods interpolate all thermo-chemical and transport properties from the generated look-up table at run time as seen in figure 2.5. However, in the GEMS implementation, only the species concentrations are updated. Because of the formulation of the energy equation in total enthalpy form, the resultant temperature and transport properties are recomputed using the species. As a result, the solved system maintains energy conservation but requires the computation of transport and thermodynamic properties from the local species composition and total enthalpy.

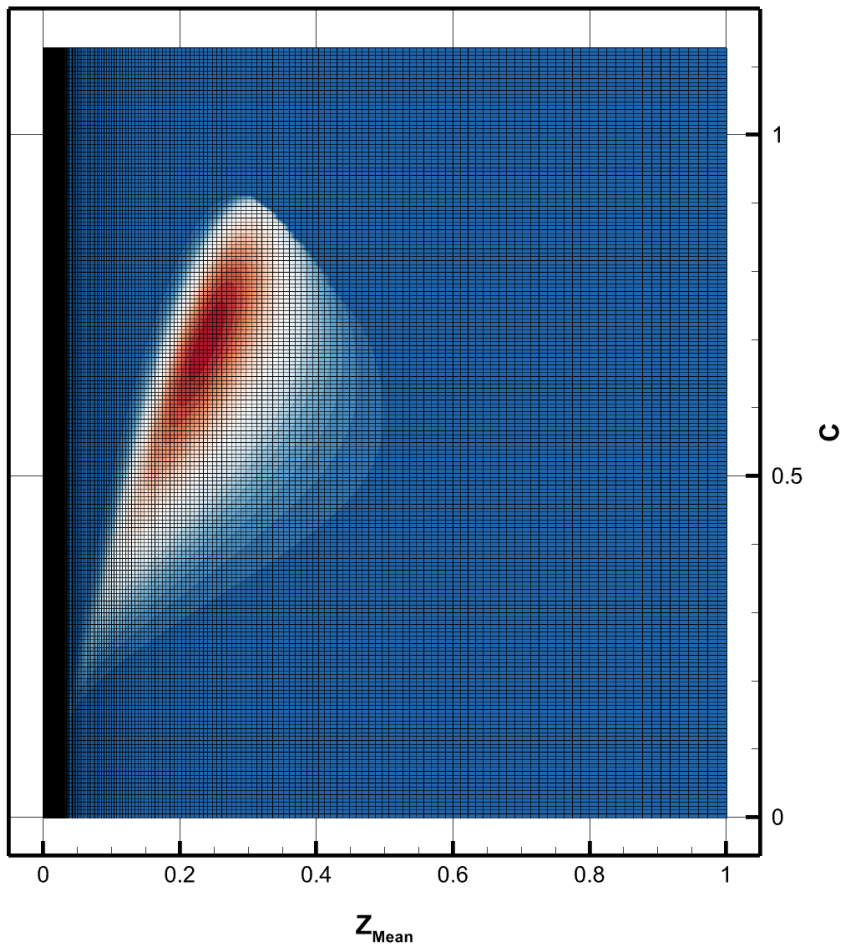


Figure 2.5: Flamelet table contour of source in progress variable $\widetilde{\dot{\omega}}_C(s^{-1})$ as a function of mixture fraction mean and progress variable for a methane-air reaction.

2.8 Time-Integration

The GEMS solver is capable of using both first and second-order accurate time derivative approximations. These are

$$\frac{\partial \mathbf{q}}{\partial t} = \frac{\mathbf{q}^{n+1} - \mathbf{q}^{n}}{\Delta t} \tag{2.70}$$

and

$$\frac{\partial \mathbf{q}}{\partial t} = \frac{3\mathbf{q}^{n+1} - 4\mathbf{q}^{n} + \mathbf{q}^{n-1}}{\Delta t}. \tag{2.71}$$

The numerical scheme used for the time-integration leverages a dual-time approach [73]. This approach introduces an additional pseudo-time derivative to the coupled equation set.

$$\mathbf{\Gamma}_p \frac{\partial \mathbf{q}_p}{\partial \tau} + \frac{\partial \mathbf{q}}{\partial t} + \frac{\partial (\mathbf{f} - \mathbf{f}_{\text{visc}})}{\partial x_i} = \mathbf{h} \tag{2.72}$$

Because $\mathbf{\Gamma}_p \frac{\partial \mathbf{q}_p}{\partial \tau} \to 0$ as $\tau \to \infty$ we can express it in terms of the primitive state without affecting the final result.

$$\mathbf{q}_p = \begin{bmatrix} p \\ u \\ T \\ Y_i \end{bmatrix} \tag{2.73}$$

Expressing to solution state in terms of primitive variables is helpful from a practical perspective. For example, computing the temperature from the enthalpy requires a Newton solve of the NASA polynomial, whereas computing the enthalpy from temperature is purely algebraic. This dual-timestep variable transformation is key to the model-form preserving least-squares with variable transformation (MP-LSVT) that is described in section 2.3.5.

2.9 Summary

This chapter introduces the governing equations used in the thesis. As a starting point, the ROM equations and methods by which a ROM is constructed from a general dynamic system. However, this type of ROM still requires the evaluation of the non-linear governing function. Therefore, this "intrusive" ROM is integrated with an existing traditional LES solver. This FOM solver is also required to generate the training data to construct the ROM. The traditional combustion modeling equations are therefore explored and detailed.

Chapter 3

Computational Scalability

3.1 Outline

The methods described in this thesis can become computationally intractable unless close attention is given to the details of scalable implementation. Computational Fluid Dynamics (CFD) solvers are generally highly tuned around the sparse ODE linear system produced by spatial numerical discretization. In contrast to that, most data-driven methods rely on dense linear algebra leveraging the entire system state at various time instances. The FOM and the ROM solution have significantly different areas of influence. In a traditional CFD solver, a single mesh points state is generally only relevant to the surrounding cells. In contrast, each mesh point has a global influence in the context of a projection-based ROM. A scripting environment can handle both paradigms(sparse and dense) for the early development of these methods. However, as this work focuses on the application of these methods to large-scale systems, a significant amount of effort needs to be spent to implement these methods efficiently.

This chapter will focus on the offline pre-processing steps needed for a projection-based reduced order model (ROM), as well as a variety of analysis techniques used in later chapters. To overcome these challenges, software is required to enable these computations at scale. While the work and tools presented may seem very matter-of-fact, almost all of the results presented later would not be possible without the development of these tools. PLATFORM in particular, can be considered an apparatus that has enabled a wide variety of studies. PLATFORM acts as an abstraction that hides most parallel computing

challenges from the user, allowing them to focus on methodological development. This ease of use has not only enabled the work within this thesis but also studies focused on aspects of ROM development [52, 53] not pursued in this thesis.

Finally, success in ROM development has proved to be an extremely nuanced endeavor. The methods used to train trial basis are extremely sensitive to centering and normalization choices. For a large-scale case a single decomposition might take many hours to compute the required decompositions and test the online portion. PLATFORM's capabilities have, in many cases, reduced this to minutes. This ability to iterate rapidly should not be underestimated and has allowed a number of best practices in projection ROM training to be developed by other users.

This chapter will include portions of work previously published in [74].

3.2 Offline Pre-Processing

As we apply the ROM equation

$$\mathbf{W}^T\mathbf{V}\frac{\mathrm{d}\hat{\mathbf{q}}}{\mathrm{d}t} = \mathbf{W}^T\mathbf{P}\mathbf{f}(\tilde{\mathbf{q}}, t). \tag{3.1}$$

The creation of the trial basis $\mathbf{V}$ is truly data-driven as $\mathbf{V}$ is almost always computed via the Proper Orthogonal Decomposition (POD) of a data matrix.

3.2.1 Proper Orthogonal Decomposition

In practice, POD is usually achieved by calculating the singular value decomposition (SVD),

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}}, \tag{3.2}$$

where $\mathbf{U} \in \mathbb{R}^{M \times min(M,N)}$ and $\mathbf{V^T} \in \mathbb{R}^{min(M,N) \times N}$ are the spatial and temporal modes respectively, with their relative weights (in descending order) described in the diagonal matrix $\mathbf{S} \in \mathbb{R}^{min(M,N)}$. In the following psuedo-code, I/O bottlenecks are shown in red, memory bottlenecks are shown in blue, and compute bottlenecks are shown in cyan.

Algorithm 1 Proper Orthogonal Decomposition

```
for k ← 1 to N do
    A[:, k]=unroll(a_k)
    U,Σ,V^T = SVD(A,'econ')
end for
for k ← 1 to N do
    u_k=reshape(U[:, k],mesh)
end for
```

Most scientific computing packages (MATLAB/Python) have SVD solvers which are very efficient. However, for large data sets, problems can arise in moving the data from its output format to its respective column and then re-associating it with the desired data format again. For our use example, we need to take mesh-associated data files and load them into the required matrix form. This data matrix can reach a significant size and becomes cumbersome and intractable on a single-machine system. Even on specialized large-memory system solutions. In short, even if you had a scripting machine capable of holding the data matrix in memory, the amount of time needed to read and write the data combined with the sheer size of the data matrix SVD would make it a poor solution.

Method of Snapshots

As a starting point, there are algorithmic methods we can use to somewhat mitigate these issues. For a majority of large-scale computation analysis, the number of snapshots (M) is expected to be much less than the degrees of freedom (N), resulting in a tall, skinny ($M >> N$) snapshot matrix. We can take advantage of this using the method of snapshots to reduce the computational load. Instead of computing the full reduced SVD of an $M \times N$ system, we can perform an eigen-solve of a symmetrized data matrix ($\mathbf{A}$) of size $N \times N$, which can have significantly lower computational cost.

$$\boldsymbol{A^T A V} = \boldsymbol{\Lambda V}. \tag{3.3}$$

When using this method, the columns of $\mathbf{V}$ of equation 3.3 are identical to the rows of $\mathbf{V^T}$ in equation 3.2 when sorted in descending order based on their corresponding eigenvalues

($\mathbf{\Lambda}$). The corresponding spatial modes can then be solved using the data matrix as,

$$\mathbf{u_i} = \frac{1}{\sigma_i}\mathbf{A}\mathbf{v_i}, \tag{3.4}$$

where $\mathbf{v_i}$ is the ith temporal mode and σ_i is the corresponding singular value.

The MOS method is shown in algorithm 2

Algorithm 2 Method of Snapshots

```
for i ← 1 to N do
    for j ← 1 to N do
        ATA[i, j] = unroll(a⃗_i) · unroll(a⃗_j)
    end for
end for
V, Σ² = eig(AᵀA)
for k ← 1 to N do
    U[:, k]= (1/σ_i) AV[:, k]
    u⃗_k=reshape(U[:, k],mesh)
end for
```

The MOS method provides computational conveniences not present in the more formal SVD. Memory-wise, The full data matrix $\mathbf{A}$ does not need to be stored. Compute-wise, the eigen-solve of an $N \times N$ matrix is usually cheaper than an SVD of an $M \times N$ system. However, when $M \to N$ the benefits of this method become negligible as the eigen-solve approaches the same complexity as the equivalent SVD while still incurring wall time costs of I/O as information from disk is needed at every loop iteration. Overall this method is a trade-off. For problems that are sufficiently tall and skinny ($M >> N$), MOS makes POD tractable in memory at a significant increase in the disk I/O requirement. As a result, a majority of offline ROM preparation for smaller problems is conducted using the MOS in a scripting environment (Python, MATLAB).

3.3 Computational Challenges

The underlying challenge in applying algorithms like those presented to large problems is composed of two competing factors. The standard SVD method reduces the overall disk I/O requirement at the cost of memory while MOS eliminates the memory requirement

while significantly increasing the disk I/O requirement.

3.3.1 Memory Problem

In an accurate 3D reacting flow simulation, one can expect meshes containing millions of elements. For a reacting flow solver, there can be upward of 8 state variables at each mesh element. Finally, robust decompositions of anywhere from 1,000 to 10,000 snapshots can be required. This leads to the resultant data matrix (as visualized in Fig. 3.1) being,

$$\boldsymbol{A} = M \times N = \begin{matrix} (\text{DOF} & \times & \text{\# of vars}) & \times & (\text{\# of snapshots}) \\ (O(10^7) & \times & O(10^2)) & \times & O(10^{(3-5)}) \end{matrix}. \tag{3.5}$$

Assuming data are being stored in double-precision floating-point arrays, the total memory cost can range up to

$$\boldsymbol{A} = \mathcal{O}(10^{(11-13)} \text{ elements}) * \underbrace{8\text{bytes}}_{\text{double precision}} = \mathcal{O}(\text{Tbytes}). \tag{3.6}$$

Even with large memory nodes available on high-performance computing (HPC) systems, scripting resources will be limited to the single node's computing capability. Here lies the fundamental trade-off of this problem. We need a solution that leverages available resources to minimize wall time while processing a very large data set.

3.3.2 I/O Problem

The second major bottleneck is loading data onto a distributed system efficiently. Even if the other challenges (computing and memory) were resolved, reading and writing data to and from the disk would dominate the wall time. This becomes especially important when we consider the solution to the memory problem, the distributed memory setup. The multiple processes will need to find and load subsets of the total system, but data reads will be non-contiguous, with multiple processes needing access to different parts of the same file.

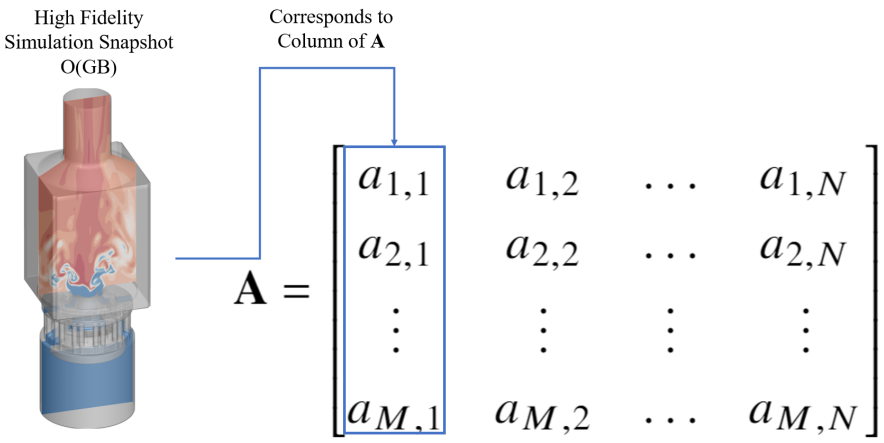


Figure 3.1: Example of organization of simulation data into a discrete matrix. The 3D field data at the t^{th} time instance is re-arranged into a 1D vector, which corresponds to one column of the data matrix **A**.

3.3.3 Objectives

As these challenges are present across all of the preliminary ROM studies a better solution was required. This software tool needs be able to:

1. Perform large-scale (e.g. on matrices with $\mathcal{O}(10^{13})$) elements) distributed memory numerical linear algebraic operations relevant to data-driven decomposition and modeling;

2. Optimize total wall-clock time of I/O operations and data transfer;

3. Enable fast prototyping and turn-around required by large-scale engineering problems;

4. Leverage resources used to generate the target data for pre- and post-processing;

5. Include adapter functions allowing the application to a variety of data formats;

The tool is designed to interface with data formats commonly used in computational engineering and allows efficient and economical data loading compatible with the distributed memory format required by ScaLAPACK. This allows users who use high-performance computing (HPC) resources traditionally used for *online* computations and

leverage those same resources to accelerate large-scale linear algebra operations (e.g. reduced-order model development, modal decomposition, and sensor placement).

This software has developed into the Parallel Linear Algebra Tool FOr Reduced Modeling (PLATFORM) software. PLATFORM should be thought of as an apparatus that has enabled a vast majority of the results presented in this thesis, in particular the application of ROMs to large problems.

3.4 Software Description

PLATFORM consists of two major components: The first, `pMat`, forms a generic wrapper abstraction of the data format required by the ScaLAPACK [75] compute routines. The second component, `meta`, acts as a translator between file formats and the distributed matrix format required by ScaLAPACK.

3.4.1 `pMat`

To process the data matrix in distributed memory, `pMat` is centered around and driven by the available parallel linear algebra tools. Both PBLAS and ScaLAPACK operate on a MPI process grid which defines how processors are organized into a geometric process grid. However, the more important aspects of the `pMat` object are the distributed matrix information, including its global dimensions, the local blocking factor, and other information used in the ScaLAPACK and PBLAS routines is contained in a *matrix descriptor*. This blocking structure's distribution is visualized in figure 3.2. The construction of these non-contiguous memory blocks is abstracted by the `pMat` class.

3.4.2 `meta`

With the memory challenges addressed by distributed memory via `pMat`, the other major challenge of I/O translation from the files to the distributed matrix. Serial methods for loading the data into memory are inadequate. As shown in the example in Fig. 3.1, we can see that usually the dataset can be loaded as contiguous columns of the desired data

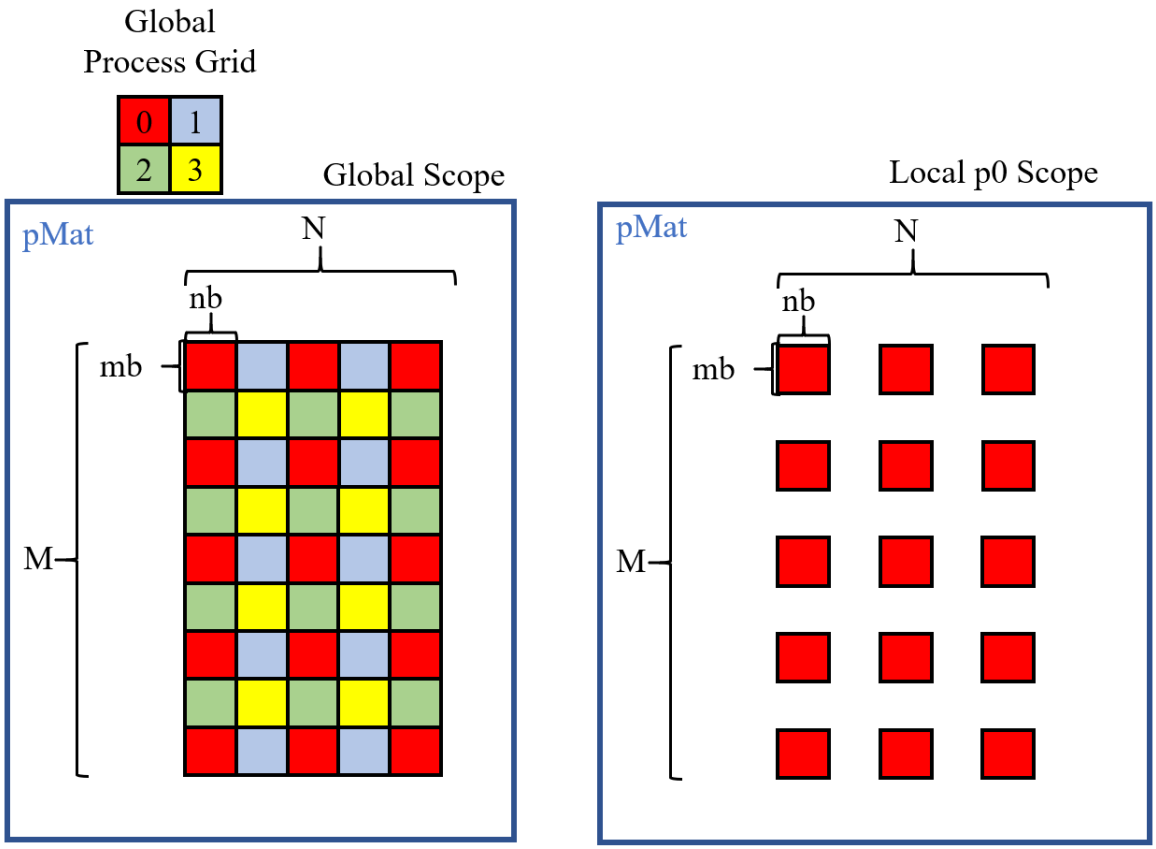


Figure 3.2: `pMat` object containing local array and blocking information.

matrix. When the data is stored in the distributed data matrix, it is effectively disassociated from the spatial domain and physical meaning. Within the `pMat` abstraction, different variables or spatial locations are simple entries in the defined matrix. `meta` keeps track of any associated `pMat` objects origins to allow the data to be re-associated for visualization or output after any manipulations have occurred.

I/O

The `meta` class is built around taking advantage of the structure of individual files generated by most simulation software. Examining the block-cyclic distribution of the matrix in Fig. 3.2, we can see that collective commands over the entire MPI communicator are not required. In fact, for the case of a perfect square grid of MPI process only a small subset of processors contain the elements of a given column (or file). To take advantage of this locality during the loading process, each "column" of processes is separated into its own sub-communicator. The "parent" process of each sub-communicator requests the file, and reads it into memory, and distributes it throughout the column process distribution. For a set of N files, the theoretical time spent reading or writing the files will be $\frac{N}{\sqrt{p}}$ where p is the total MPI rank count. As a result, an "embarrassingly parallel" speed up of the file I/O is achieved. A schematic of this process is shown in figure 3.3. Additionally, because a smaller subset of processors is accessing each file, the demand placed on the cluster file system is significantly reduced.

3.4.3 Dataset Metadata

In addition to controlling data I/O, the `meta` class also keeps track of the details of what data is stored in the distributed matrix. For example, if the columns of `pMat` are a combination of pressure, temperature, and density values, the `meta` abstraction encodes this information. When reading or writing data stored in a specialized file format, `meta` will pass this information to generate or recall the structure of those file types from the distributed data.

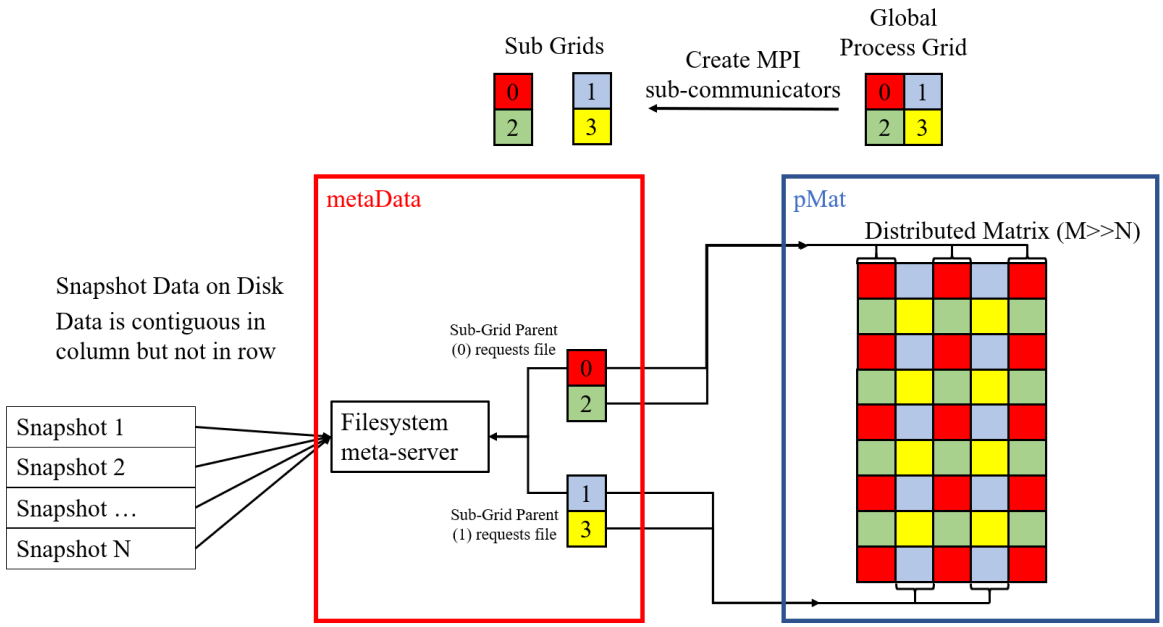


Figure 3.3: Example I/O strategy for loading snapshots into memory distributed over four process.

3.5 Data Formats Supported

The PLATFORM framework supports a variety of different data input and output formats, which include dumping the desired data matrix to a single binary file, parsing individual snapshot files corresponding to columns of the data matrix, and a serial-based parser that maps a specialized data format onto the standard `meta` wrapper. The code is designed to handle two primary data interfacing strategies.

1. Large matrix binary file
    - The individual physical snapshots are pre-processed as a full group and the desired matrix is dumped into a single large file.
    - Easy to implement, fast I/O, large single files are preferred by cluster file systems
    - Significantly increases data footprint, limited ability to modify problem-specific data
2. Set of files

(a) Binary set

- Similar format to large binary file, but individual columns of the data matrix are split by file
- Allows for faster I/O, but for extreme cases, the number of files can become a limiting factor

(b) Format-specific

- Allows problem-specific information to be extracted from a file with the same Pros and Cons of binary set option

3.6 Performance

It must be noted that for this framework while performance is appreciated the overall priority is making the problem tractable in memory with an I/O strategy that does not unduly hinder execution. However, profiling can ensure the scaling of the I/O routines is as expected. Scaling was tested on the ERDC Onyx cluster, configured with standard nodes consisting of two Intel Broadwell E5-2699v4 for a total of 44 physical cores per node utilizing the Cray Aries interconnect. More importantly, each node has 121 GBytes of physical memory for this application. Strong scaling was tested using 16 nodes and the results are shown in Fig. 3.4. A global matrix was written and read for each test using MPI-I/O and `meta`. In Fig. 3.4 we show the scaling of the different I/O methods described above. Here we observe the rough $N/\sqrt{p}$ scaling for the "batch" file I/O. This is to be expected based on the I/O strategy shown in Fig. 3.3 where the total communicator grid is split based on the corresponding file columns. Compared to a single file MPI-IO call, we see significant improvement in speed, particularly in reading. This is primarily due to the batch option taking advantage of the known structure of the file set. This allows the data read to be contiguous compared with MPI-I/O. However, it is expected that for very large file sets ($> 1e6$), the number of file queries may strain a cluster file system retrieval server. So for extreme edge cases, the MPI-IO single file system may be required.

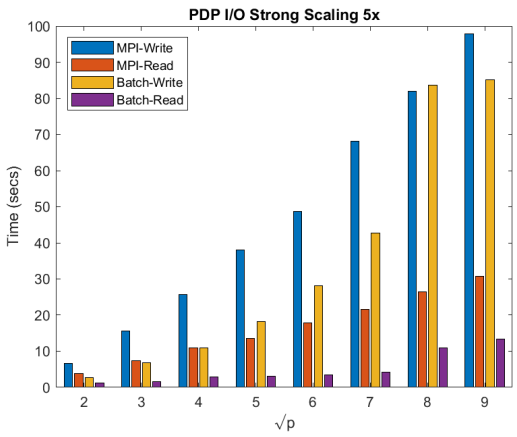


Figure 3.4: PLATFORM I/O routine strong scaling(M=1e7 N=p*5) comparison.

3.7 Example Usage

While the performance and overall structure of the PLATFORM code are essential in enabling ROM pre-processing, it is more important to remember how the abstraction eases use. Various examples and tutorials were developed as a part of the PLATFORM documentation, but we highlight two here that are most relevant to the ROM workflow.

3.7.1 Matrix Allocation

Matrix allocation of a dense matrix in a distributed setting is challenging to keep track of global indices (the rows and columns of the global matrix) and local indices (the rows and columns of the memory allocated on the local processor). Using raw ScaLAPACK requires significant expertise to allocate the required memory buffers and descriptors. An example of PLATFORM matrix allocation takes the form

```
1 int M = 2000;
2 int N = 100;
3 // initialize distributed matrix of size MxN with double-precision
     zeros
4 pMat A(M, N, 0.0);
```

Listing 3.1: Matrix initilization

The `pMat` constructor will automatically determine the appropriate matrix blocking factor and initialize the matrix descriptor and the local memory block. These additional

descriptors are required in determining each processor's local scope and parameters for the ScaLAPACK and PBLAS driver routines as shown in Listing 3.3.

Listing 3.2: Matrix initialization standard output

```
initializing Cblacs
Processor Grid M(cols)=2 N(rows)=2
local processor row 0 and column 0

Creating Matrix
M=2000 N=100
mb|nb = 50
Mat is Double
local elements 50000 of 200000 global elements

Descriptor type: 1
BLACS context: 0
Global Rows: 2000
Global Cols: 100
Row Blocking factor: 50
Column Blocking factor: 50
Process row where first row is: 0
Process Col where first col is: 0
Leading Dimension: 1000
Memory usage(data only) MB = (negligible <1 MB)
```

3.7.2 Driver Interfaces

In addition to `pMat`'s responsibility for creating and maintaining the distributed arrays, it also contains wrapper functions for ScaLAPACK routines, such as the SVD. The corresponding ScaLAPACK driver function for double-precision real-valued numbers is `pdgesvd`, shown in Listing 3.3.

```
void pdgesvd(
    // compute singular vector flags input
    const char * JOBU, const char *JOBVT,
```

```
    // problem Size input
    const int *M, const int *N,
    // A matrix input
    const double *A, const int *IA, const int *JA, const int *DESCA,
    // singular value output
    const double *S,
    // left singular vector output
    const double *U, const int *IU, const int *JU, const int *DESCU,
    // right singular vector output
    const double *VT, const int *IVT, const int *JVT, const int *DESCVT
   ,
    // workspace info
    const double *WORK, const int *LWORK, const int *ierr);
```

Listing 3.3: ScaLAPACK `pdgesvd` C interface

These inputs can introduce additional development time for those unfamiliar with ScaLAPACK and distributed matrices. Additionally, the SVD routine requires allocating workspace memory based on the matrix size and distribution. In PLATFORM, most of these parameters and prerequisites are collapsed into simple `pMat` driver interfaces, which reduce end-user workload. An example case of the abstraction, applying the SVD to a matrix, is shown in listing 3.4.

```
int M = 2000;
int N = 100;
// init MxN matrix with zeros
pMat A(M, N);

//load in expected data

// initialize U, VT, and S
pMat U(M, min(M, N));
pMat VT(min(M, N), N);
vector<double> S(min(M, N));
A.svd(A.M, A.N, U, VT, S);
```

Listing 3.4: Example driver function using `pMat`

Such abstractions reduce the requirement of a user's knowledge of distributed memory operations to make the framework more user-friendly and flexible to extend to any application. Among others, the SVD (`p?gesvd`), general dense matrix multiplication (`p?gemm`), and QR decomposition (`p?geqpf`) have been added to `pMat`. (here `?` can change based on the operation precision, double vs. float) In addition to the simple wrappers of existing ScaLAPACK routines, drivers can be easily implemented to achieve other operations. For example, PLATFORM can calculate the pseudoinverse, a combination of the SVD and a sequence of outer products. This calculation is critical in a variety of model reduction techniques.

3.8 Summary

PLATFORM was developed primarily to enable a variety of work that requires high-memory dense linear algebra. Its applications within this thesis include:

1. The offline pre-processing steps needed for a projection-based reduced order model (ROM) (ubiquitous across all chapters).

2. The leveraging of this pre-processor for more general data-driven analysis tools (Primarily used for the DMD analysis in chapter 4).

3. The integration of the dense linear algebra and load balancing capabilities into a traditional sparse CFD solver. (Discussed in detail in Chapter 6).

Chapter 4

Dual-Swirl Gas Turbine Combustor

4.1 Outline & Introduction

Gas turbine model combustors (GTMC) have been a mainstay in laboratory investigations of gas turbine combustor design and representative physical phenomena. The Dual-Swirl Gas Turbine Model Combustor (GTMC), developed by Meier et al. [16] at the German Aerospace Center (Deutsches Zentrum für Luft-und Raumfaht(DLR)), is a particular combustor of this type. The past and present literature of this burner was described in the introduction. Portions of the work presented in this chapter were previously published [76]. As described in chapter 1, the goal of projection-based ROMs is to provide the same capability in prediction as LES-type models at a fraction of the computational cost. In this chapter, a traditional LES study of a gas turbine model combustor (GTMC) is conducted and used as the full-order model (FOM) to construct a projection-based reduced-order model (ROM).

4.2 Experimental Setup

The burner configuration is shown in Fig. 4.1. The setup consists of a combustion chamber with a square cross-section connected to a single plenum via a fixed dual swirler assembly. Dry atmospheric air is fed to the lower cylindrical plenum. The two swirlers are fed by this common plenum with symmetric piping to the upper swirler. The lower swirler feeds into a central converging nozzle (diameter of 15mm at nozzle termination), while the

	Flame A	Flame B
$\dot{m}_{air}(g/min)$	1095	281
$\dot{m}_{fuel}(g/min)$	41.8	12.3
$P_{th}(kW)$	34.9	10.3
ϕ_{global}	.65	.75

Table 4.1: GTMC operating conditions.

upper swirler feeds a co-annular diverging section (outer diameter 25mm, inner diameter 17mm at burner face). The outer annular section smoothly contours to the bottom plane of the combustion chamber. Un-swirled methane fuel is injected between the two swirled air streams co-axially. The fuel injection ports are formed by stacking the two swirler plates and fed by three radial fuel lines. The central nozzle terminates 4.5 mm below the burner face. The combustion chamber is composed of a square cross-section (85 mm^2) with chamfered post corners with a total height of 110 mm. The chamber terminates by contracting to a cylindrical chimney (40mm diameter), opening to the laboratory atmosphere. The walls in the experimental setup are quartz to allow for optical access with substitute walls with mounted probes. A representative flow field is shown in Fig. 4.2.

Two steady-state operating conditions are the target of this work. These two conditions, referred to as "flame A" and "flame B" respectively, are shown in table 4.1. Flame A represents a higher flow rate stable flame, while flame B is a more unstable flame shown to exhibit a thermoacoustic instability at 290 Hz. In the experimental literature, a third flame C was also considered; however, this operating condition was designed as a flame with intermittent lift-off and flashback. In the scope of this work, we focus on the flame A and B operating conditions.

4.3 Full-Order Model

In the context of the reduced-order modeling pursued in this work, the accuracy of the full-order model relative to experimental data is not strictly relevant. However, from a modeling perspective, it is desirable to show that the reduced-order model in replicating the FOM would be able to validate well against experimental data. ROMs have not been

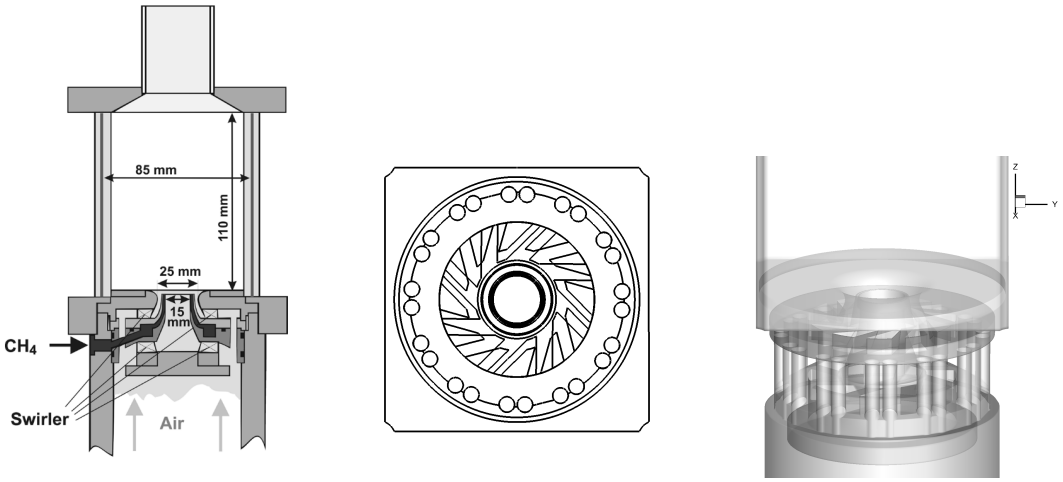


Figure 4.1: Burner schematic [6] (Left), stacked internal cutaway of swirler geometry(Center), and external iso-surface (Right).

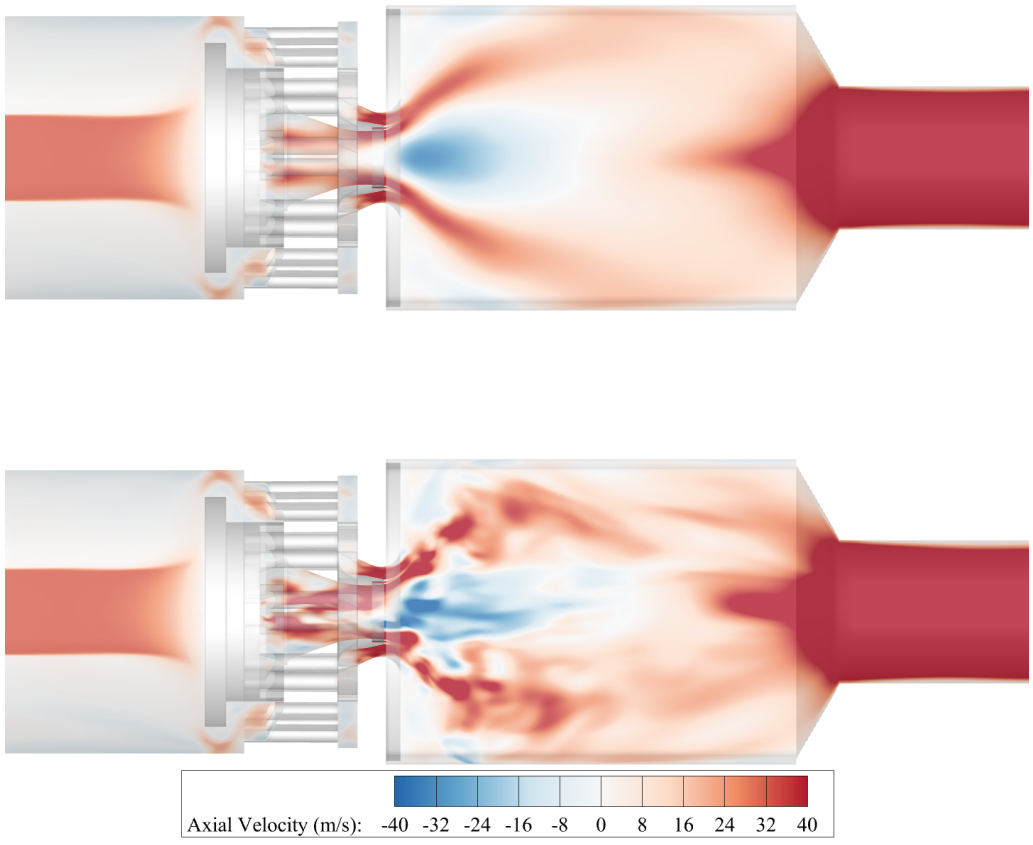


Figure 4.2: Representative averaged (Top) and instantaneous (Bottom) axial velocity fields for the flame A configuration.

pursued on problems of this scale and complexity. Performing the simulation study as it would occur in a traditional LES sense helps identify problems arising from the scale of the data and LES solver.

4.3.1 Computational details

The Large-Eddy Simulations (LES) were conducted using an in-house CFD code, the General Equations and Mesh Solver (GEMS) [59]. GEMS is a message passing interface (MPI)-based parallel, second-order accurate in time and space finite volume solver. An implicit dual time-stepping scheme is used, with Roe fluxes [77] and the Barth-Jespersen flux limiter [78]. The numerical robustness of the solver allows for the resolution of near-wall flow features while using high aspect ratio grids. The simulation was conducted on Engineer Research and Development Center (ERDC) Onyx on 2200 cores. Each Onyx node is composed of two 22-core Intel E5-2699v4 Broadwell processors for a total of 44 cores with 128 GBytes per node running on the Cray Aries interconnect. The case was run with a physical time-step of one microsecond. The flame A simulation was run for a total of 200 milli-seconds of real-time, accounting for approximately two hundred characteristic flow-through periods of the combustion chamber or 70 periods for the entire domain. The approximate wall time per flow-through period of the combustion chamber was two hours.

The mesh comprises a fully structured multi-block topology with 7.5 million hexahedral cells. The computational domain is shown in Fig 4.4. The lowest minimum angle of hex elements is 24 degrees, which is limited by the geometry (the angle at which the lower swirler vanes intersect with the core nozzle). Finally, the point spacing was allowed to vary to limit the lowest quality elements to areas where reactions do not occur, such as the plenum, upper combustion chamber, and chimney. As shown by previous works [25, 23] the modeling of both the swirling vanes and air plenum of the combustor is critical for accurate prediction. Because of this, the entire combustor geometry is modeled. The only significant geometric modeling simplification was the fuel injection nozzle. The 72 individual fuel injector holes were approximated as a circular slot with

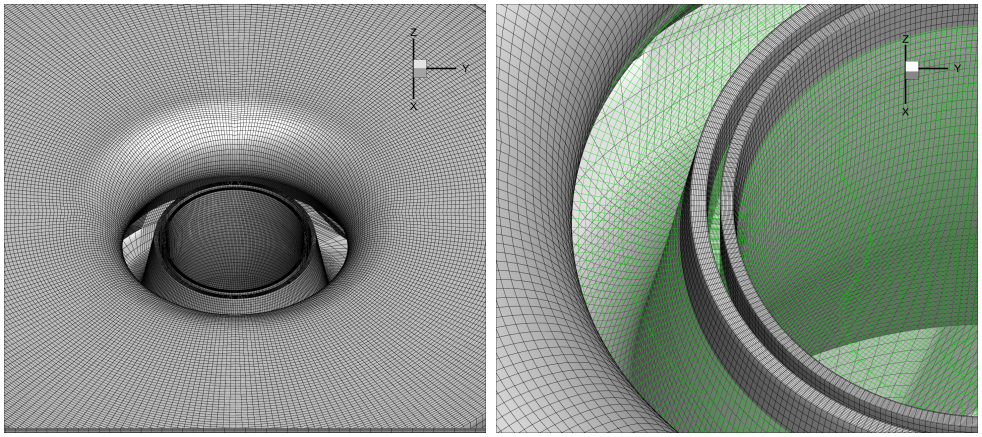


Figure 4.3: Fuel injection detail (Left) and internal cell spacing at injector face shown in green (Right).

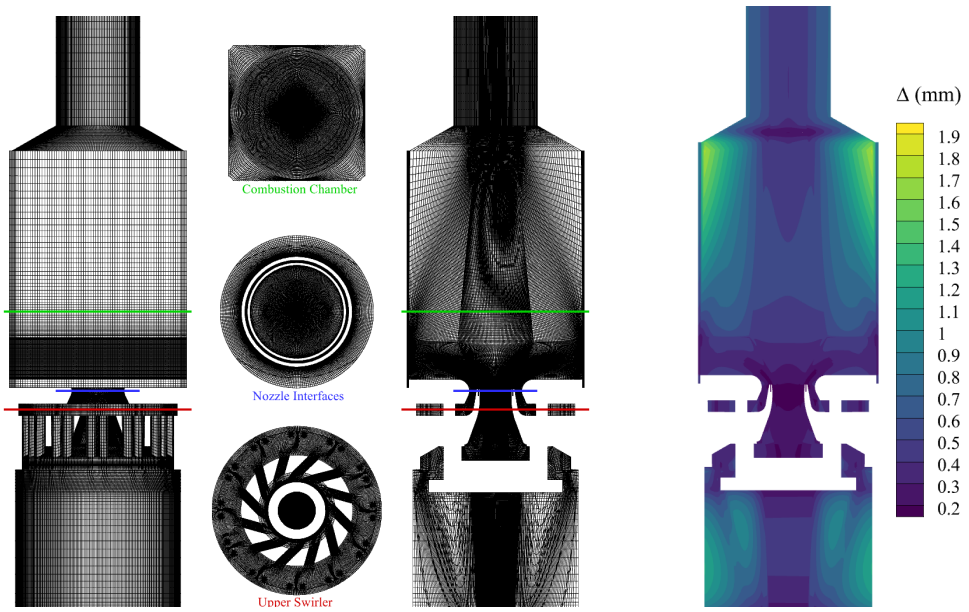


Figure 4.4: Mesh schematic with selected slice locations (Left)and mesh mean filter width slice (Right).

a matching area to accommodate the structured mesh (Fig. 4.3). The air and fuel inflow conditions were specified to match the experimental mass flow. In systems utilizing swirler vanes, the swirler geometry is predicted to generate nearly all of the significant turbulence. With this consideration, both inlets were specified as uniform velocities. The entire fuel plenum consists of 3 feed lines arranged radially (Fig. 4.1). These were not fully meshed; the fuel boundary condition is recessed from the approximated slot. The outflow boundary condition was specified as a constant pressure set to atmospheric conditions. The wall boundaries were set to be adiabatic with an enforced no-slip condition.

The overall mesh topology focused on clustering cells in the lower combustion chamber. The minimum mesh spacing of $\Delta = 0.1$ mm was constrained by the smallest feature of the system, the fuel injector, resulting in ten cells across the fuel injector ports (Fig. 4.3). A snapshot of the resulting mesh Δ is shown Fig. 4.4. Fine cells are clustered in the nozzle and lower combustion chamber regions. Because of the requirements of multi-block meshing and a desire to optimize computational cost, some locations have relatively coarse cells. However, these are limited to the upper corners of the combustion chamber, chimney, and lower plenum.

4.3.2 Averaged Velocity Field

The time and root mean square (RMS) averaged velocity fields are examined as an initial comparison between the simulation and experiment. A set of unsteady time realizations, representing 100 characteristic flow-through periods, were used to compute the cell-wise RMS and time averages. The resulting flow-fields are displayed for axial (Fig. 4.5), radial (Figs. 4.6), and swirl velocity (Figs. 4.7) with the corresponding experimental PIV measurement layered on the upper half of the contour. For a quantitative comparison, profiles are extracted for the experimental and computational data sets at various axial heights.

Flame A ($\phi = 0.65$)

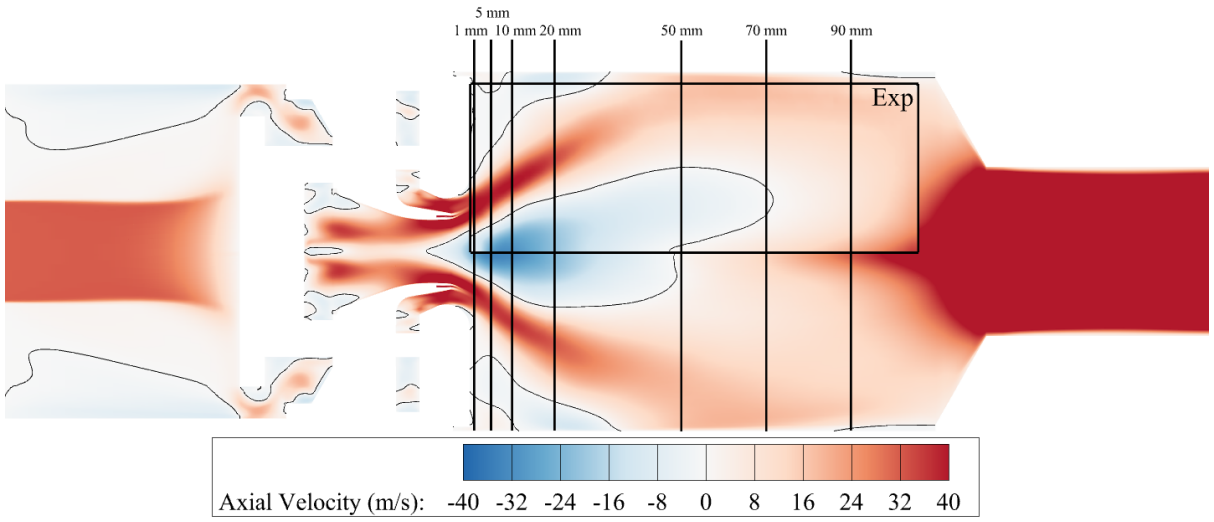
The overall structure is observed in all three dimensions with two areas of discrepancy. These variations are most observable in the axial velocity (Fig. 4.5 which has a contour line placed at the axial stagnation velocity). The first is the modeled height of the inner recirculating zone (IRZ), predicted at 4 mm lower than found experimentally. This height difference can be observed qualitatively from the contour and quantitatively in the $h = 1$ mm profile near the center line. This underprediction is expected in swirl-stabilized flows and is consistent with previous computational studies [25]. This characteristic is due to inadequate wall modeling near the nozzle termination. The second deviation is the prediction of the tail-like structure of the recirculation bubble. In works focused on modeling

a vortex breakdown structure, a significantly finer mesh resolution is required. Interestingly, despite not correctly capturing the double tail structure of the recirculation bubble, the center-line stagnation point matches the experimental observations. Examining the extracted profiles, one can observe strong agreement in both time and RMS-averaged quantities. Generally, because of the lower and smaller recirculation bubble, velocities were computed as lower than experimental values in the near nozzle region and overestimated in the combustor's upper region. Finally, the outer recirculation zones (ORZ), the recirculation zone in the lower outer corners of the chamber, are predicted accurately compared with the experimental values' size, shape, and magnitude.

A majority of the observations in the axial velocity hold when examining the corresponding radial (Fig. 4.6) and tangential (Fig. 4.7) velocity fields. Both time and RMS-averaged quantities correlate favorably with a relatively minor discrepancy in the average radial velocity profile peak, which persists in all near-nozzle profiles. This is expected to be due to inadequate wall spacing of the diverging nozzle section leading to an underestimation of the radial velocity measurements due to the slightly slimmer inner recirculation bubble.

Flame B ($\phi = 0.75$)

In contrast to flame A, flame B averaged quantities significantly differ from the experimental observations. Examining axial velocity, we observe similar deviations as flame A but to a more significant degree. The outer re-circulation zones (ORZ) are still predicted quite accurately. However, the inner circulation zone (IRZ) is under-predicted quite severely in both height from the nozzle and overall size. This is most observable in the stagnation contour and profile comparisons in axial velocity (Fig. 4.8). Examining the radial velocity, we see large deviations as the radial velocity is consistently underpredicted. This consistent over-prediction of height and underprediction of the width of the IRZ is likewise suspected to be due to inadequate wall modeling of the diverging section. Because the mass flow and resulting velocities for flame A are up to three times larger than flame B, it is suspected that for flame A the separation occurs relatively close



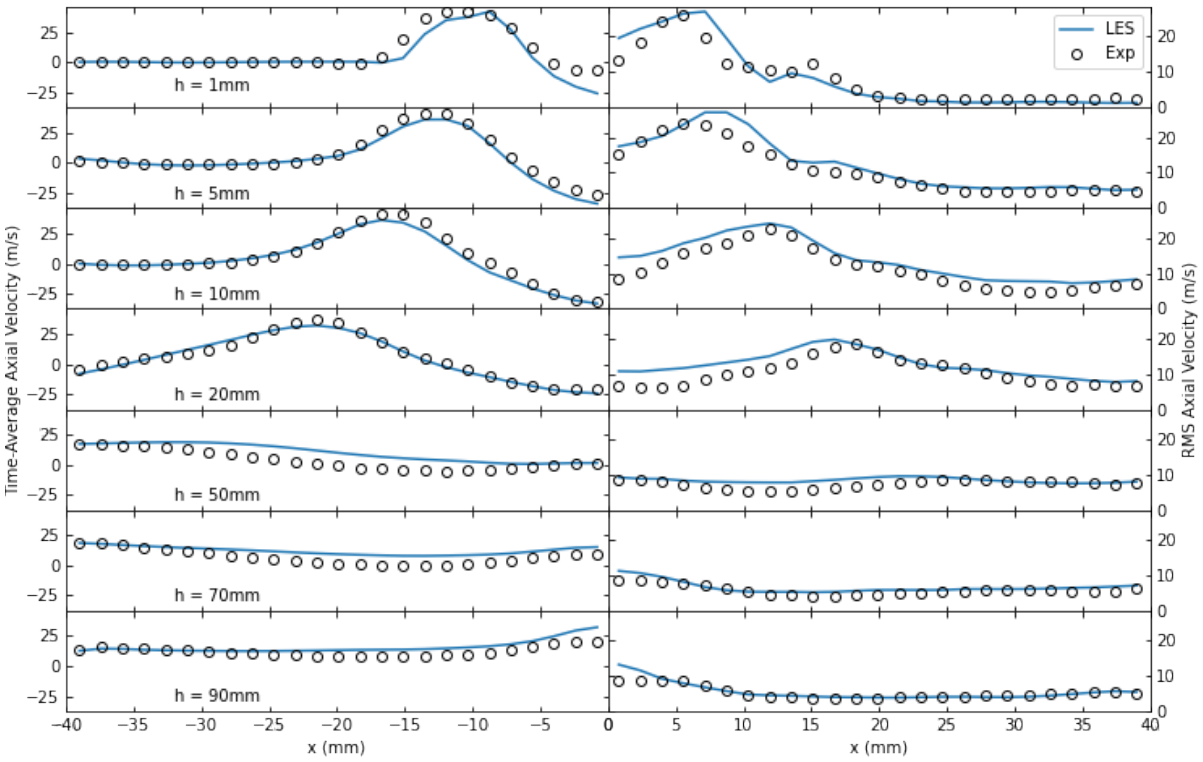
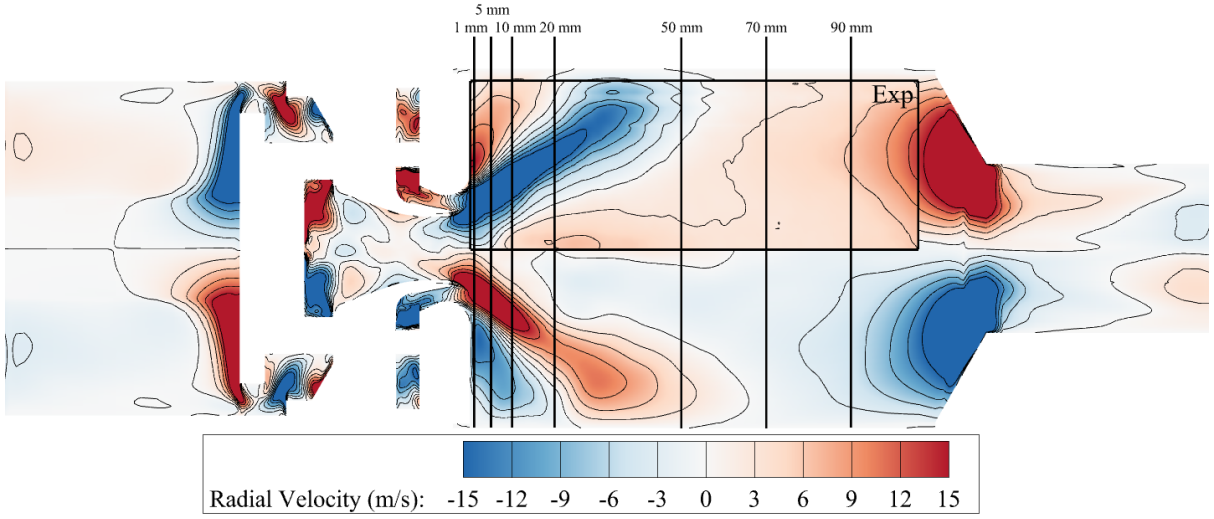


Figure 4.5: Time and RMS-averaged axial velocity comparisons: experimental data superimposed on upper half of combustor with lines at axial velocity equal to zero for the Flame A configuration.



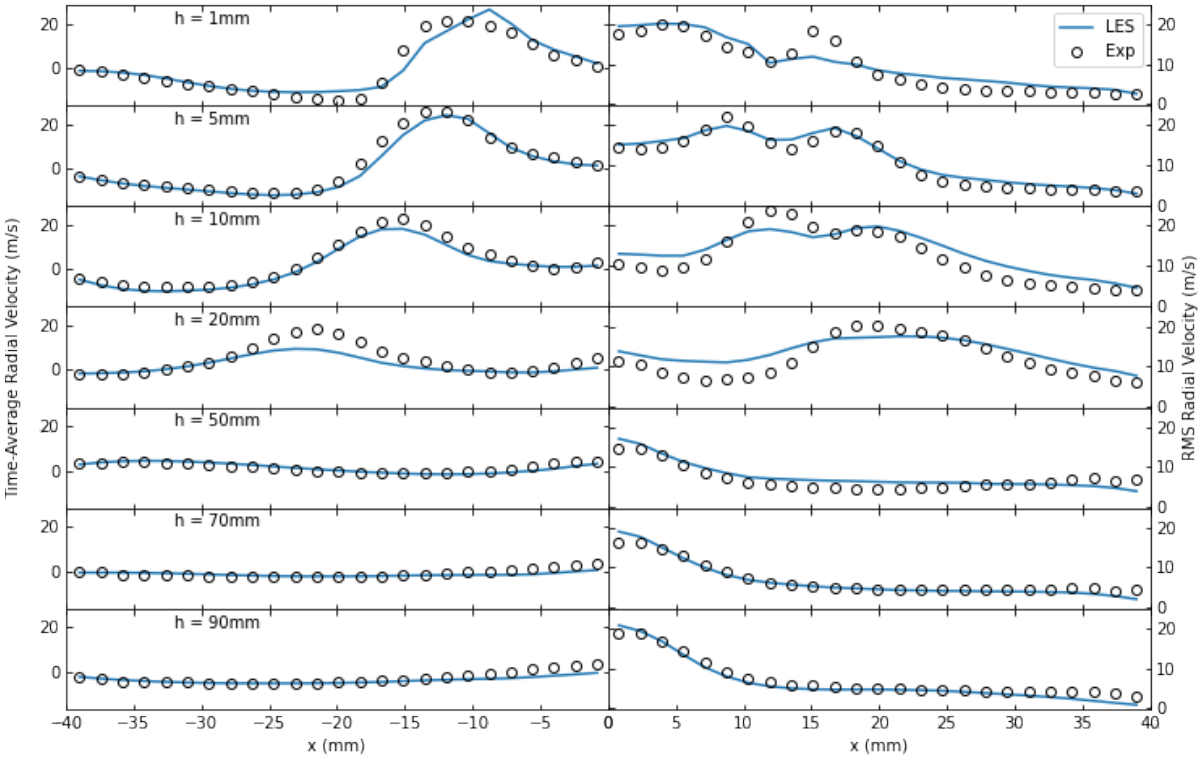
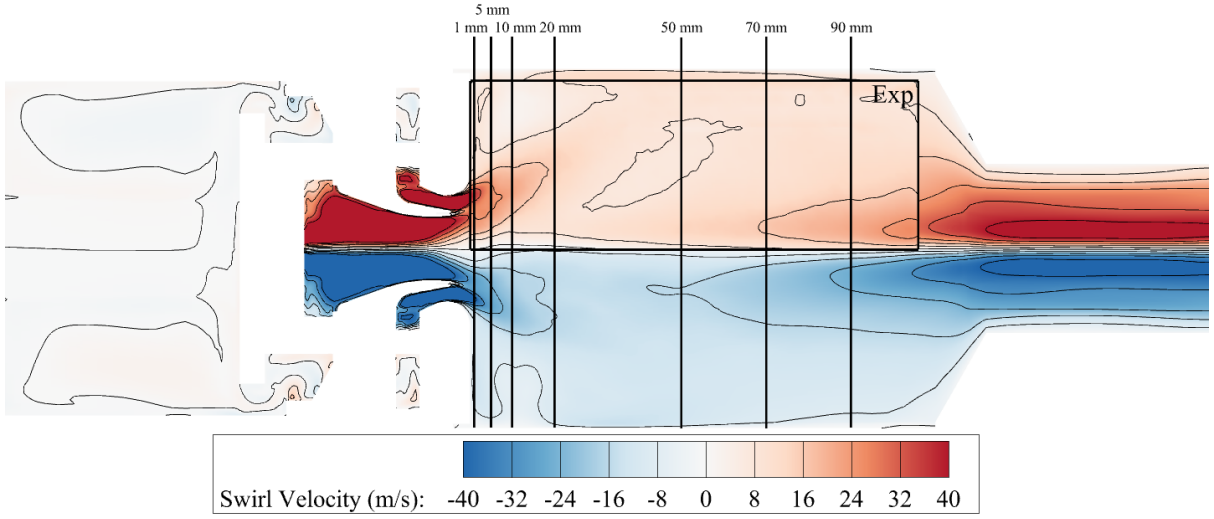


Figure 4.6: Time and RMS-averaged radial velocity for the flame A configuration.



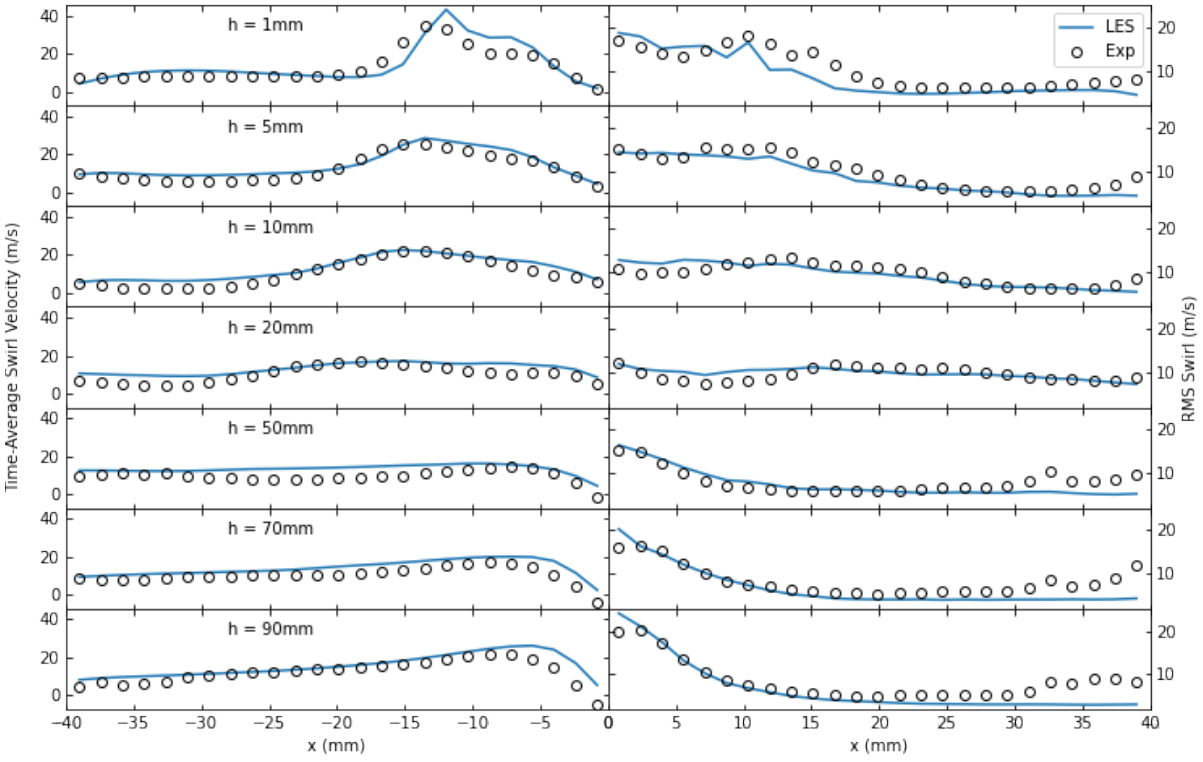


Figure 4.7: Time and RMS-averaged tangential velocity for the flame A configuration.

to the beginning of the diverging section. However, the separation point for flame B can be much farther down the diverging nozzle. In the predicted velocity field, the separation occurs much farther upstream than the experiment leading to the taller, slimmer predicted IRZ. This may be compounded by combustion modeling effects that will be discussed later.

4.3.3 Averaged Temperature and Mixture Fraction

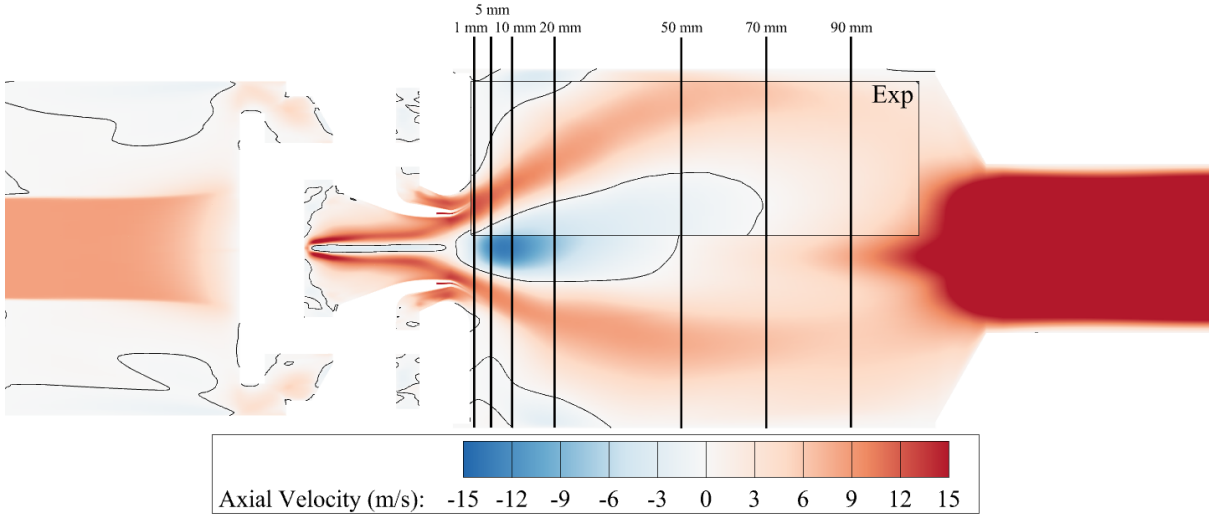
For the dual-swirl GTMC, the two conditions, flame A and flame B represent significantly different conditions [6]. Flame A corresponds to an acoustically broadband stable V-shaped flame, while flame B corresponds to a self-excited thermo-acoustically unstable case with a flat flame.

Flame A ($\phi = 0.65$)

The mid-plane contours from the flame A case is shown in Fig. 4.11 along with extracted profiles from the simulation data compared with experimental measurements. We should note that the experimental spectroscopy values are spatially averaged, which can produce deviations compared with CFD-filtered quantities. Qualitatively, flame A shows the characteristic V-shaped structure and compares well with the experimental profiles. A caveat is that the height of the flame, like the IRZ, is underpredicted. This underprediction can be observed in the relative overprediction in temperature near the burner face. This under-prediction is most likely a combination of the nozzle wall modeling, adiabatic wall boundary conditions, and partially premixed reactions. However, the overall characteristics of the temperature field are well captured.

Flame B ($\phi = 0.75$)

For the flame B operating condition, the temperature contours (figure 4.12) show a flatter flame compared to flame A; however, not nearly to the degree that is experimentally observed. Examining the temperature profiles, the center-line temperature is significantly overpredicted near the burner face by about 300 K. Additionally, the points far from the



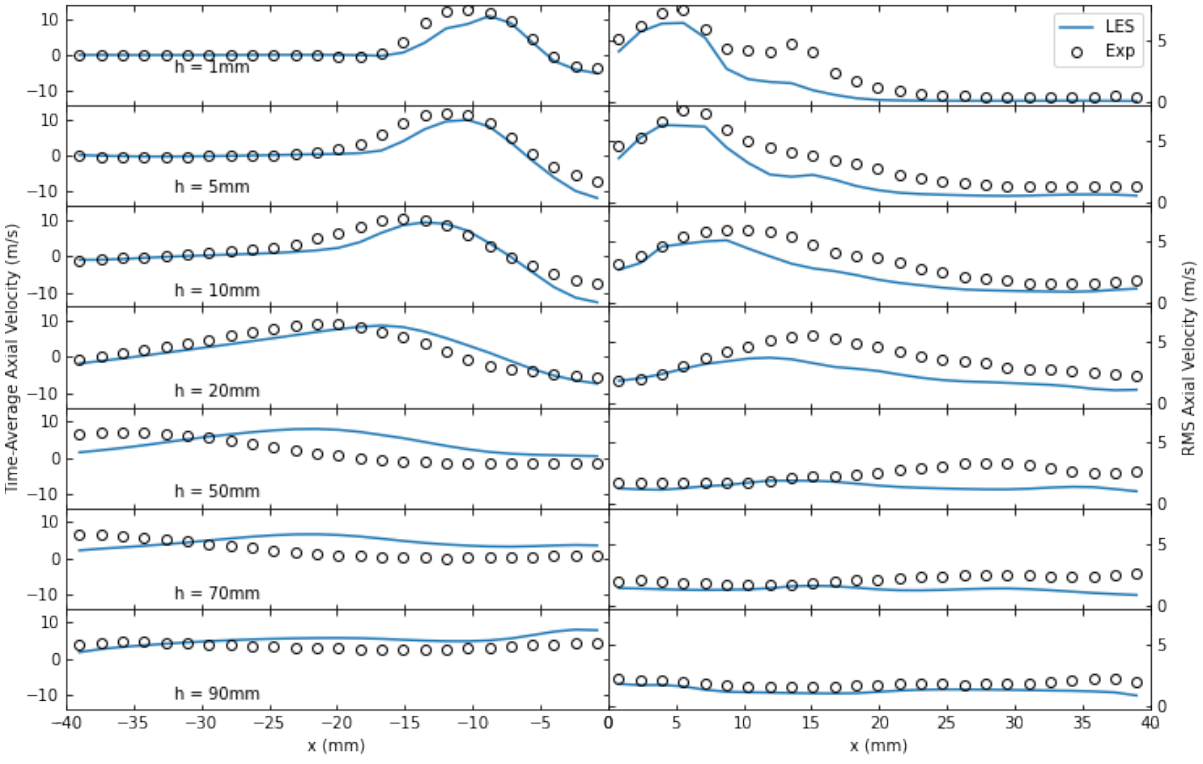


Figure 4.8: Time-averaged (left) and RMS (right) axial velocity of flame B.

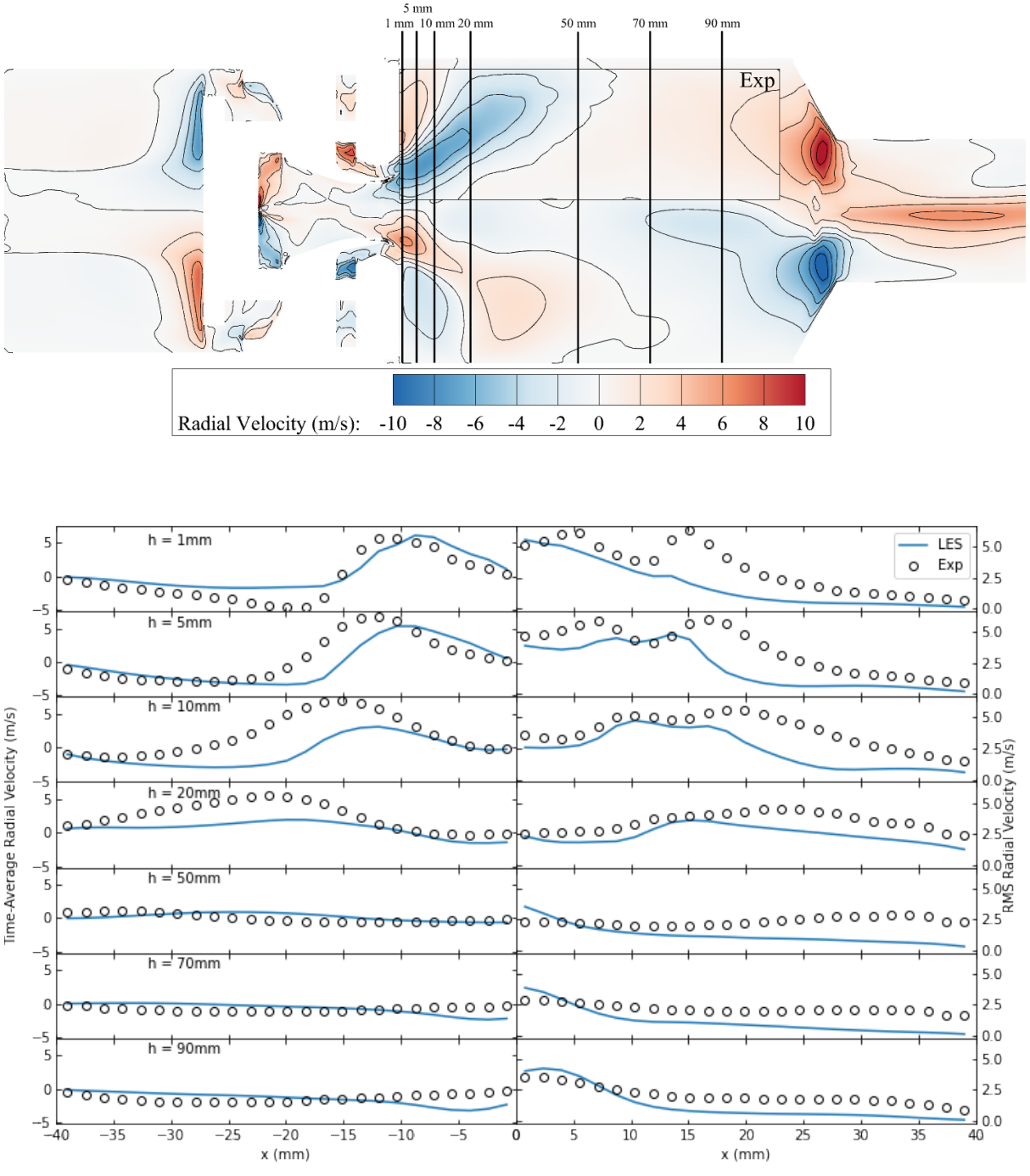
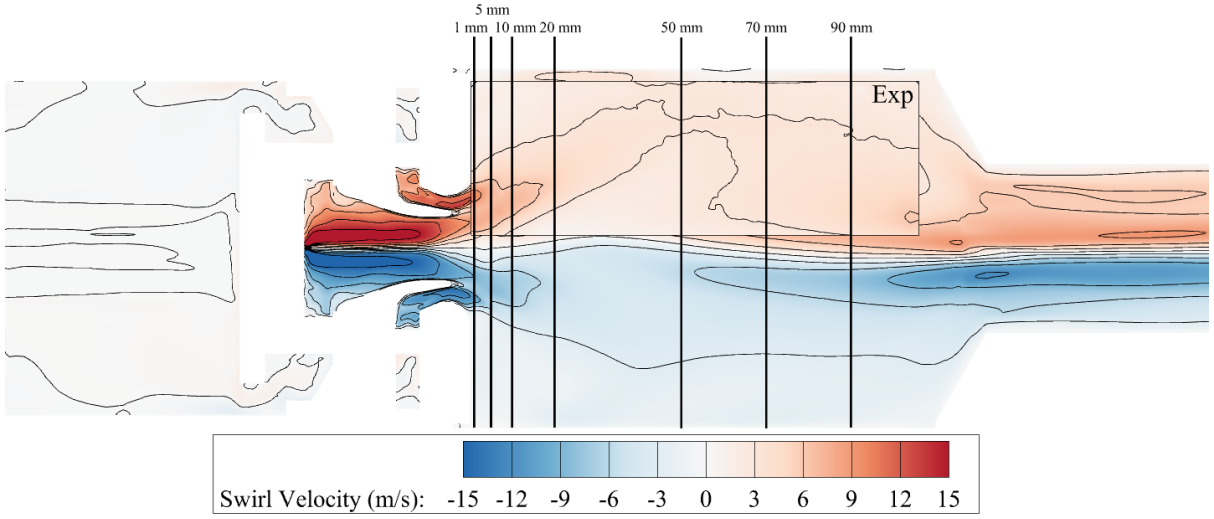


Figure 4.9: Time-averaged (left) and RMS (right) radial velocity of flame B.



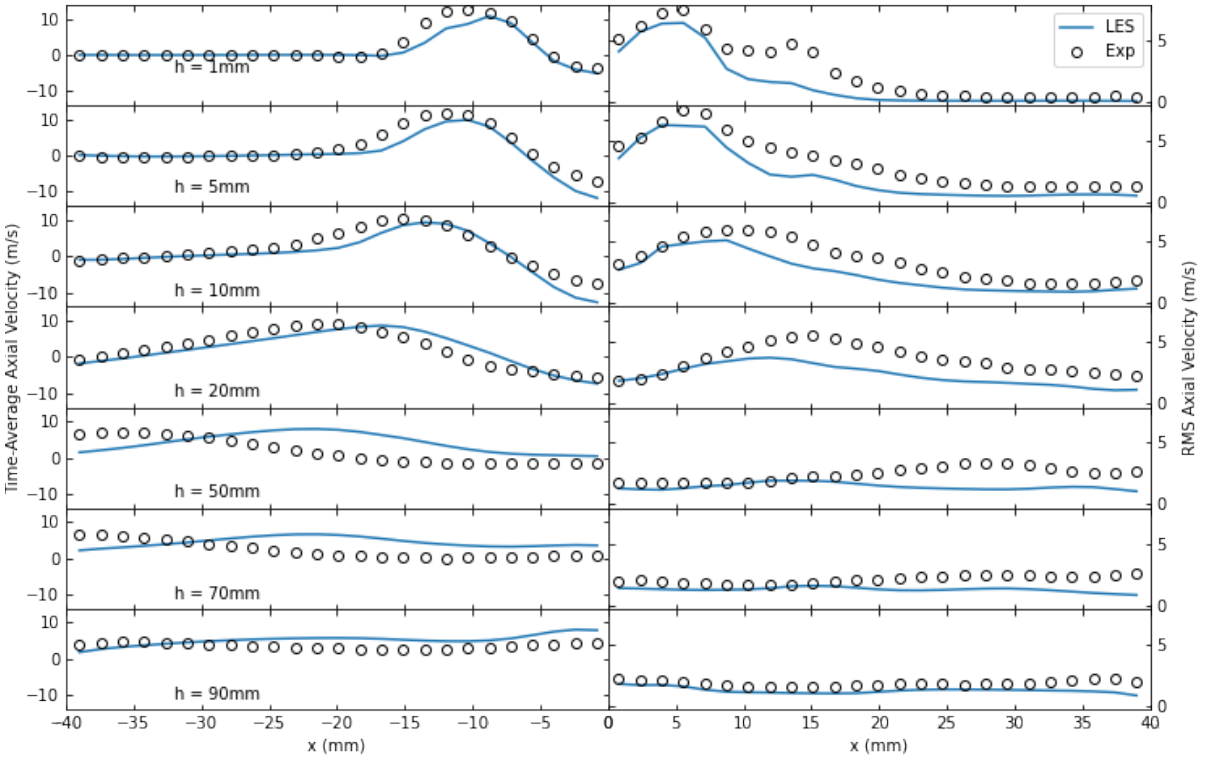


Figure 4.10: Time-averaged (left) and RMS (right) tangential velocity of flame B.

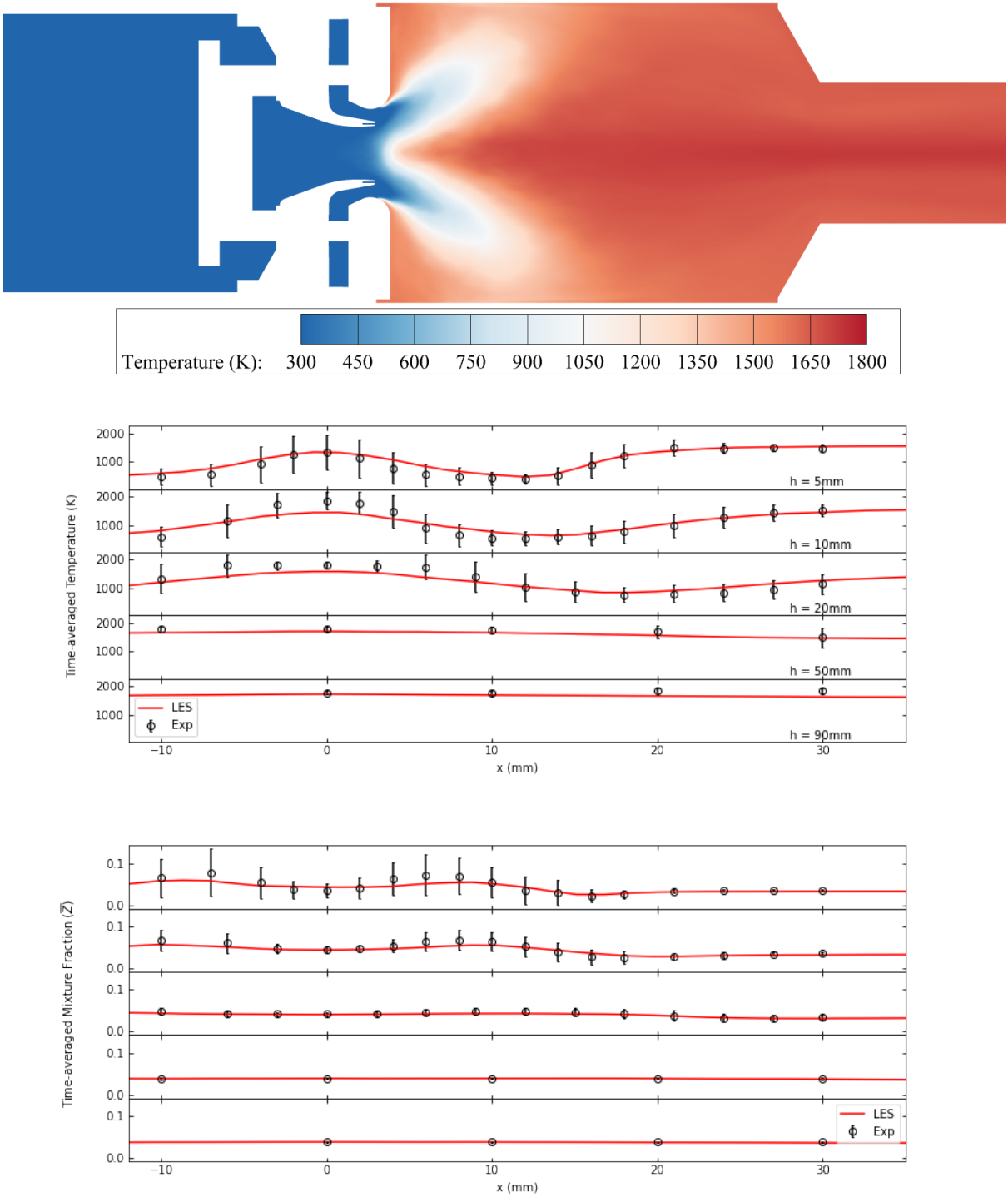
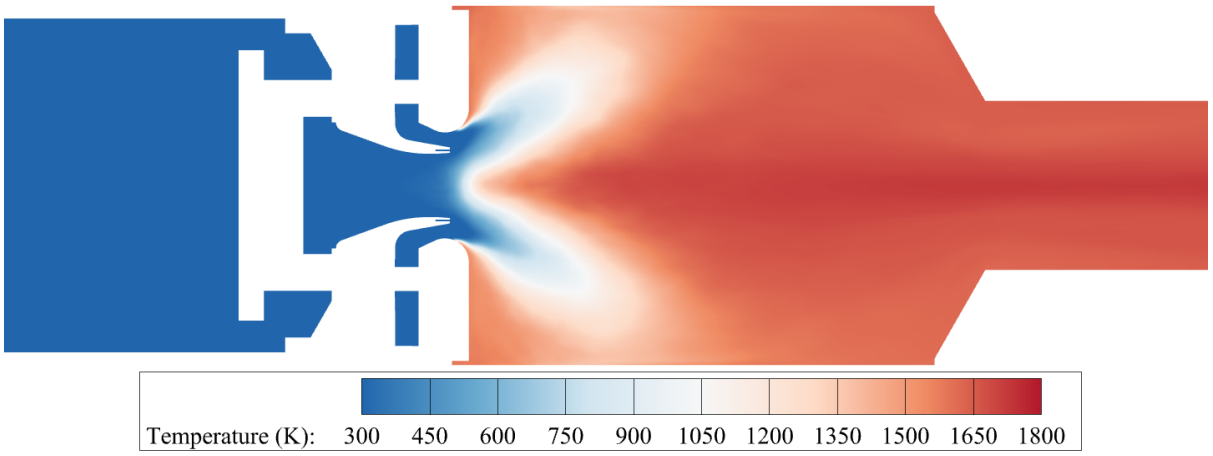
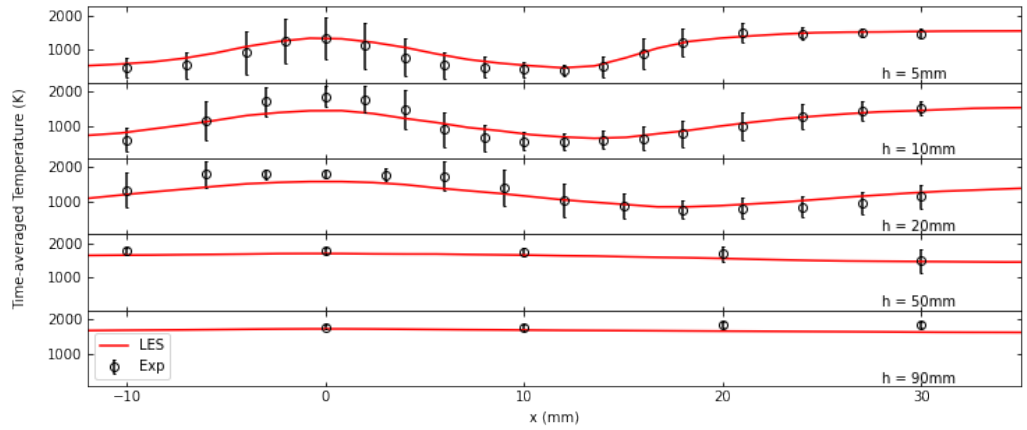


Figure 4.11: Time-averaged temperature and mixture fraction field for flame A.





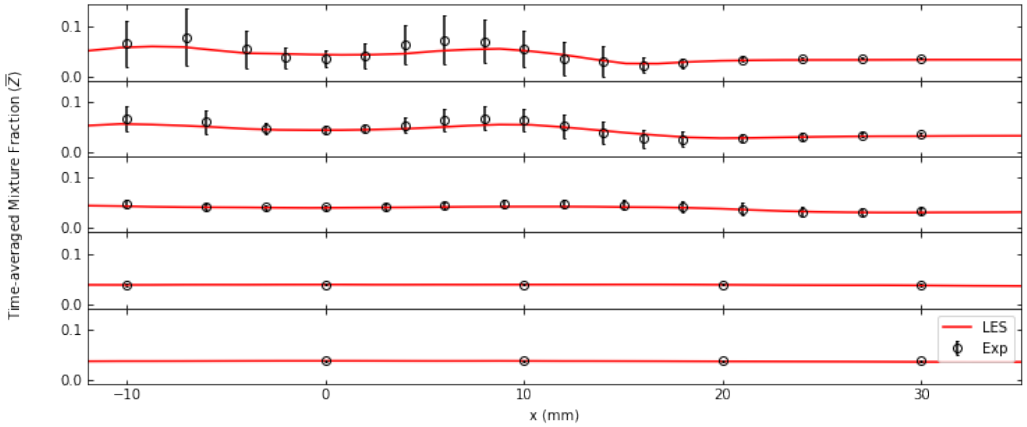


Figure 4.12: Time-averaged temperature (Top) and mixture fraction field (Bottom) for flame B.

center line (20-30 mm) are significantly over-predicted near the burner face. This is suspected to be due to the adiabatic wall conditions. Kraus et al. [18] showed that a significant amount of heat is conducted from the burner face to the plenum oxidizer streams. However, accurate material modeling of this effect is very challenging. This effect is more pronounced in the flame B case, as the flat flame naturally leads to higher temperatures near the burner face. Additionally, Chen [26] noted that modeling of the fuel plenum pipe improved the prediction for flame B.

4.3.4 Mixture Fraction Temperature Correlation

The Flamelet Progress Variable (FPVA) model does not have additional source terms to account for partially premixed flame conditions. Examining the correlation between mixture fraction and temperature in the simulation vs. experimental spectroscopy measurements can give quantitative insight into the consistency between the experimental configuration and the simulation's combustion model. This correlation for Flame A and B is shown in figure 4.13 and figure 4.14, respectively. The LES values are Favre-filtered values, while the experimental points are single-shot measurements. This filtering leads to a much wider spread in the LES mixture fractions across all locations and is particularly observable in 0-2 mm range. This is consistent with previous computational efforts of this condition [25]. There are many points at low temperatures at various mixture fractions for both flame A and B, which indicates a partially premixed flame. Despite this, flame A shows relatively strong agreement. The higher temperature clusters (0-2 and 21-27 mm) represent the IRZ and ORZ, respectively, and are regions that are mostly fully reacted products. Flame B, in comparison, features an ORZ region that is more reactive, with a large temperature range, which is not captured by the LES other than those confined to the adiabatic curve. However, as mentioned in previous sections, the flame height in both cases is under-predicted, leading to a non-negligible error in the direct comparison at $h = 5$ mm distributions. Despite no modeling terms for the premixed regions of the flame, good agreement is found in flame A with some deficiencies in the more premixed flame B. This is expected because of flame B's lower mass flow rate; the

relative convection of the flow is reduced compared to the diffusive effects (identical for both flames).

4.3.5 Unsteady PIV Comparison

While the simulation of the averaged quantities is relevant for design point characterization, in the context of thermo-acoustic instability, it gives little insight into the underlying mechanisms. For the DLR combustor, unsteady characterization was achieved using a high-speed kilohertz PIV measurements. The key hydrodynamic and acoustic frequencies can be isolated by taking specific points in the flow field. These measurements were conducted at 10 kHz for Flame A. This region is shown in figure 4.15 and is centered at the nozzle exit to examine the recirculation zones and shear layers present. These three points of interest are the swirling JET, the inner shear layer (ISL), and the inner recirculation zone (IRZ).

The comparison between the spectrum of the axial velocity probes of interest are shown in Fig. 4.15. Flame A's primary frequency is characteristic of the precessing vortex core (PVC), an unsteady helical structure characteristic of swirling flows. This structure is observed in the JET and ISL probes as they are far enough upstream for this structure to develop. The IRZ probe, on the other hand, sits just above the burner's face. For the JET and ISL probes, the PVC frequency and amplitudes are consistent at about 1700 Hz. For the IRZ probe, however, we observe relatively broadband noise across the frequency spectrum. Noting an earlier observation that the LES is under-predicting the average recirculation bubbles height by about 4 mm, a second LES spectrum is added with that offset which shows agreement much more in line with the experimental probe.

4.4 Dynamic Mode Decomposition

While single-point probes can directly compare experimental results, the nature of simulation data allows more comprehensive methods to be used. In particular, when trying to predict the characteristic frequencies of large-scale structures, the resulting spectrum

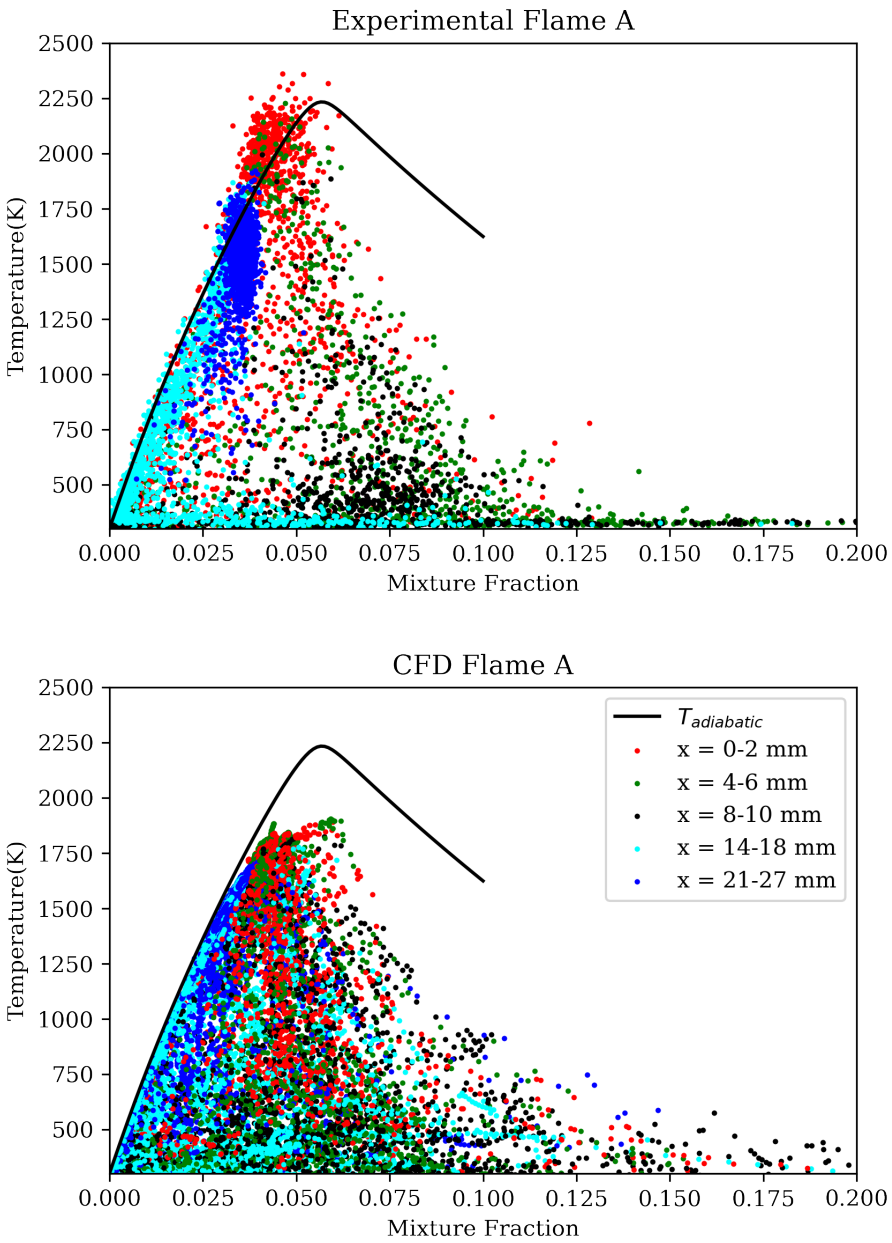


Figure 4.13: Temperature vs mixture fraction scatter plots for experimental and CFD at $h = 5$ mm for flame A.

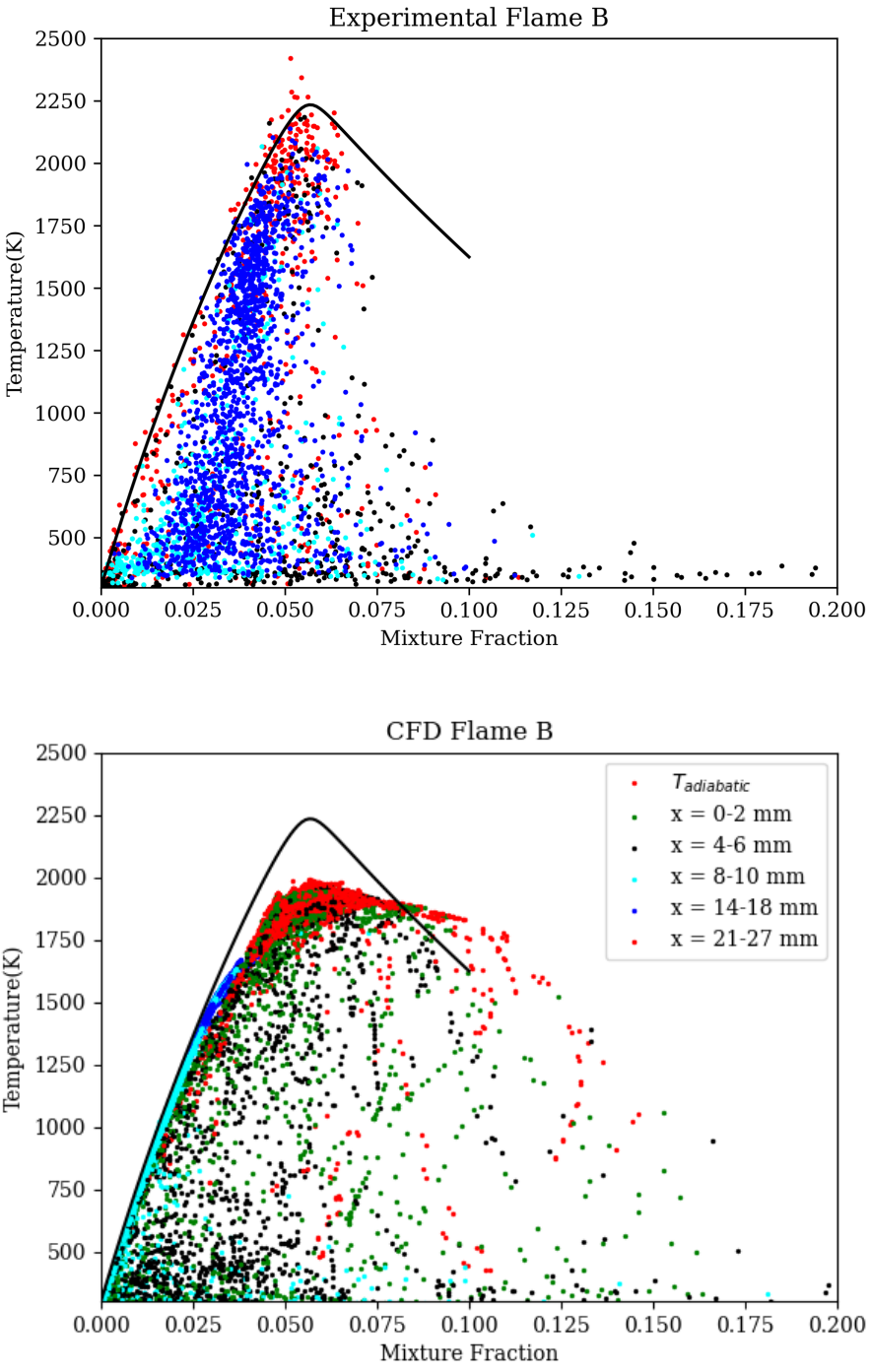


Figure 4.14: Temperature vs mixture fraction scatter plots for experimental and CFD at $h = 5$ mm.

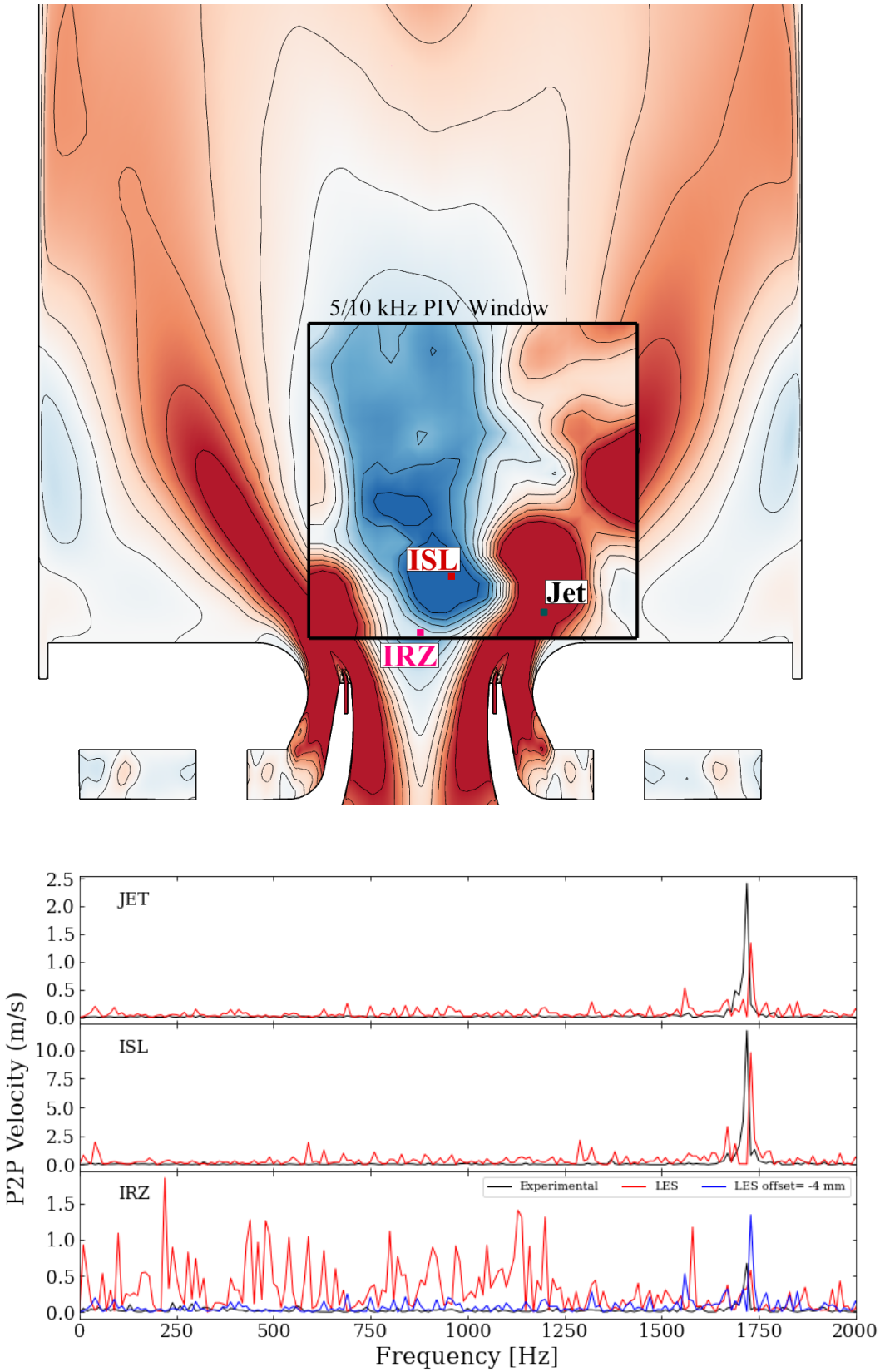


Figure 4.15: Schematic of kHz PIV window with points of interest label (Top) power spectrum of axial velocity of flame A comparison of points of interest (Bottom).

can be extremely sensitive to the probe location. Modal decompositions are a family of methods that can gain additional insight by leveraging spatial data. Boxx [79] calculated proper orthogonal decomposed(POD) modes which were able to visualize the PVC structure. It was later shown [80] that for systems exhibiting combustion instability, Dynamic Mode Decomposition (DMD) [81, 82] can create more temporally consistent mode shapes, which can both identify hydrodynamic and acoustic features. DMD extracts more physical representations of complex spatio-temporal structures than POD by constraining the temporal modes to discrete frequencies. Details of this methodology for sampling choice and formulation can be found in previous works [80, 83] but is summarized here.

This family of model decomposition methods begins by organizing the field data into vectors of a snapshot matrix.

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_N \end{bmatrix} \in \mathbb{R}^{M \times N} \tag{4.1}$$

We define the data matrix and time advanced data matrix as,

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_{N-1} \end{bmatrix} \in \mathbb{R}^{M \times N-1} \tag{4.2}$$

$$\mathbf{A}_2 = \begin{bmatrix} \mathbf{a}_2 & \mathbf{a}_3 & \dots & \mathbf{a}_N \end{bmatrix} \in \mathbb{R}^{M \times N-1} \tag{4.3}$$

For stability we form the similarity form of the time advancement matrix $\mathbf{S}$, which advances the data matrix A_1 in time to A_2

$$\mathbf{A}_2 = \mathbf{A}_1 \mathbf{S}, \tag{4.4}$$

as

$$\tilde{\mathbf{S}} = \mathbf{U}\mathbf{A}_2\mathbf{V}\mathbf{\Sigma}^{-1}, \tag{4.5}$$

where

$$\mathbf{A}_1 = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T. \tag{4.6}$$

In a similarity form, the eigendecomposition of $\tilde{\mathbf{S}}$ approximates that of $\mathbf{S}$.

$$\tilde{\mathbf{S}} = \tilde{\mathbf{T}}\tilde{\mathbf{\Delta}}\tilde{\mathbf{T}}^{-1}. \tag{4.7}$$

The spatial modes (DMD modes) will be constructed as

$$\mathbf{\Phi} = \mathbf{U}\tilde{\mathbf{T}}. \tag{4.8}$$

Using general decomposition form, we form the temporal modes $\mathbf{Y}$ as

$$\mathbf{Y}^{T-1} = \mathbf{V}\mathbf{\Sigma}^{-1}\tilde{\mathbf{T}}, \tag{4.9}$$

The response corresponding to that mode can be computed as

$$\mathbf{R_i} = \psi_i \mathbf{y}_i^T. \tag{4.10}$$

Applying this methodology to the simulation data, one can visualize the complex 3D hydrodynamic features such as the PVC more effectively. Further, this analysis is more robust to discrepancies in probe location as it is applied to the entire spatial field. In addition to the core methodology, additional prepossessing using singular value truncation and total-least squares (TLS) preconditioning as described by Kutz et al. [84]. This prepossessing is significant in applying this combustor as the instability amplitudes are extremely small (0.1% of mean pressure) and can be easily diluted by noise, especially with the statistical limits imposed by the CFD run time.

4.4.1 Decomposition of high-frequency PIV measurements

DMD is conducted on the small high-frequency PIV measurements described earlier and compared to the LES data set interpolated onto an identical mesh to compare the experimental and computational decompositions directly. The same sampling rate and the total number of snapshots used corresponded to the available simulation data. The resulting spectra are shown in Fig. 4.16. Similar to the point analysis, good agreement was found between the computation and experimental results. The PVC frequency is well identified

in both the interpolated LES and experimental data sets. Otherwise, the LES contains a slightly higher energy level across the overall frequency space. This is suspected to be due to the reflection of acoustic energy from the pressure outlet boundary condition.

Applying the same methodology to the non-interpolated 3D CFD dataset, we can extract a 3D representation of this structure's procession (Fig. 4.17), confirming the existence of the PVC in the full-order model.

4.5 Reduced-Order Modeling (ROM)

At this point, we have conducted a traditional LES study of a complex combustor design and validated the flame A case against experimental data. We now will attempt to construct a projection-based ROM using the data generated for the flame A operating condition.

4.5.1 Basis Generation

We now describe the procedure for generation of the desired trial basis $\mathbf{V}$. For the reacting flow problems exhibited in this work, the primitive solution vector is organized as

$$\mathbf{q}_p = \begin{bmatrix} \mathbf{p} & \mathbf{u} & \mathbf{T} & \mathbf{Z_m} & \mathbf{Z''^2} & \mathbf{C} \end{bmatrix}^T. \tag{4.11}$$

Each of the N solution snapshots gathered by evolving Eqn. 2.5 in time are stored as columns of the complete data matrix as shown in Eq. 4.11,

$$\mathbf{Q} = \begin{bmatrix} \mathbf{q}(t_1) & \mathbf{q}(t_2) & \dots & \mathbf{q}(t_N) \end{bmatrix}. \tag{4.12}$$

Next, the reference state $\overline{\mathbf{q}}$ (e.g. the initial condition snapshot $\mathbf{q}(t_0)$ or a time-averaged field) is subtracted from each snapshot, and the resulting matrix is normalized by the diagonal matrix $\mathbf{P}$,

$$\mathbf{Q}' = \mathbf{P}(\mathbf{Q} - \overline{\mathbf{q}}\mathbf{1}^T), \tag{4.13}$$

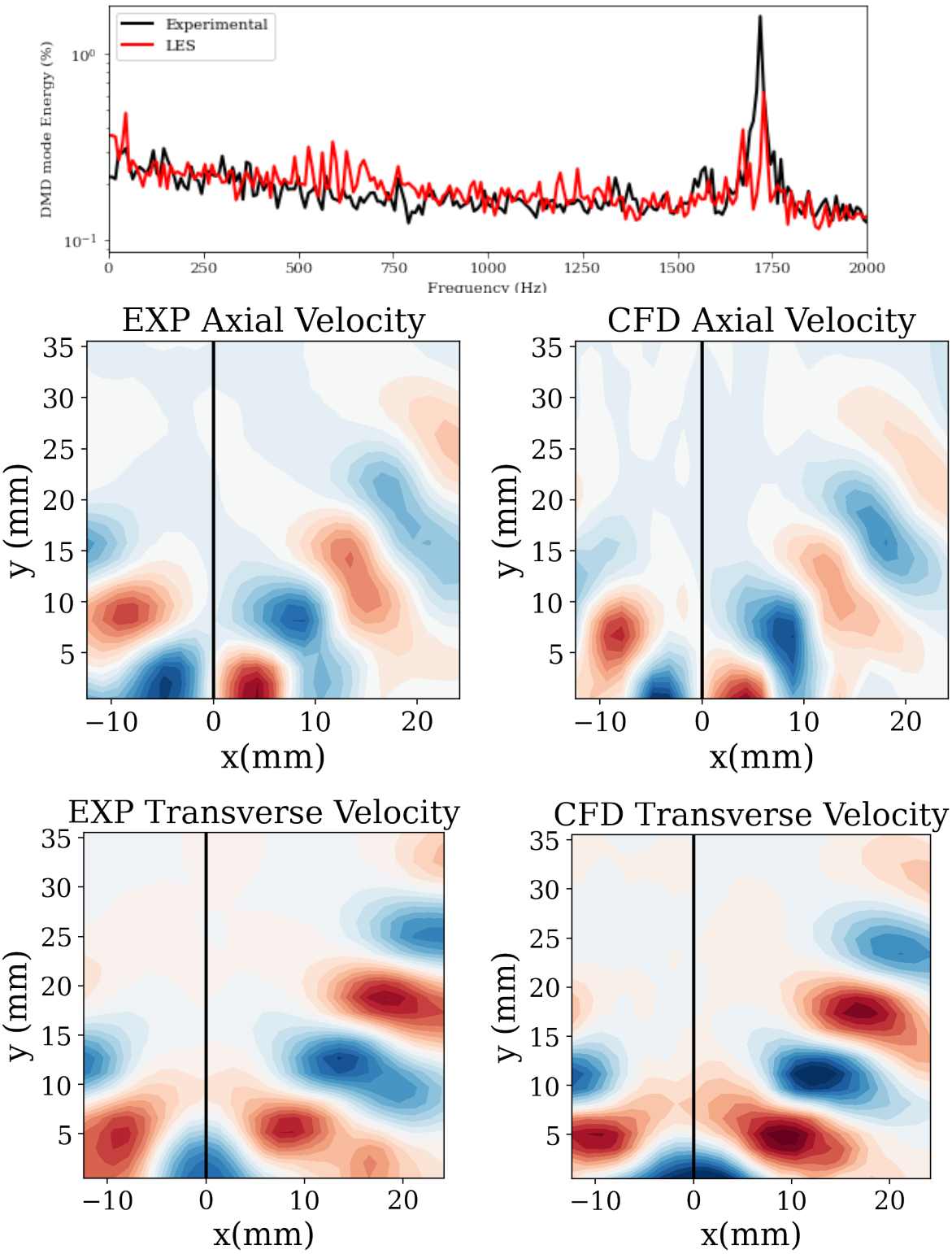


Figure 4.16: DMD spectrum of flame A axial velocity of the PIV data (Top) compared with interpolated CFD data for flame A with mode shapes corresponding to PVC peak visualized for axial velocity(Middle) and transverse velocity (Bottom). Note the experimental window is offset from the center line.

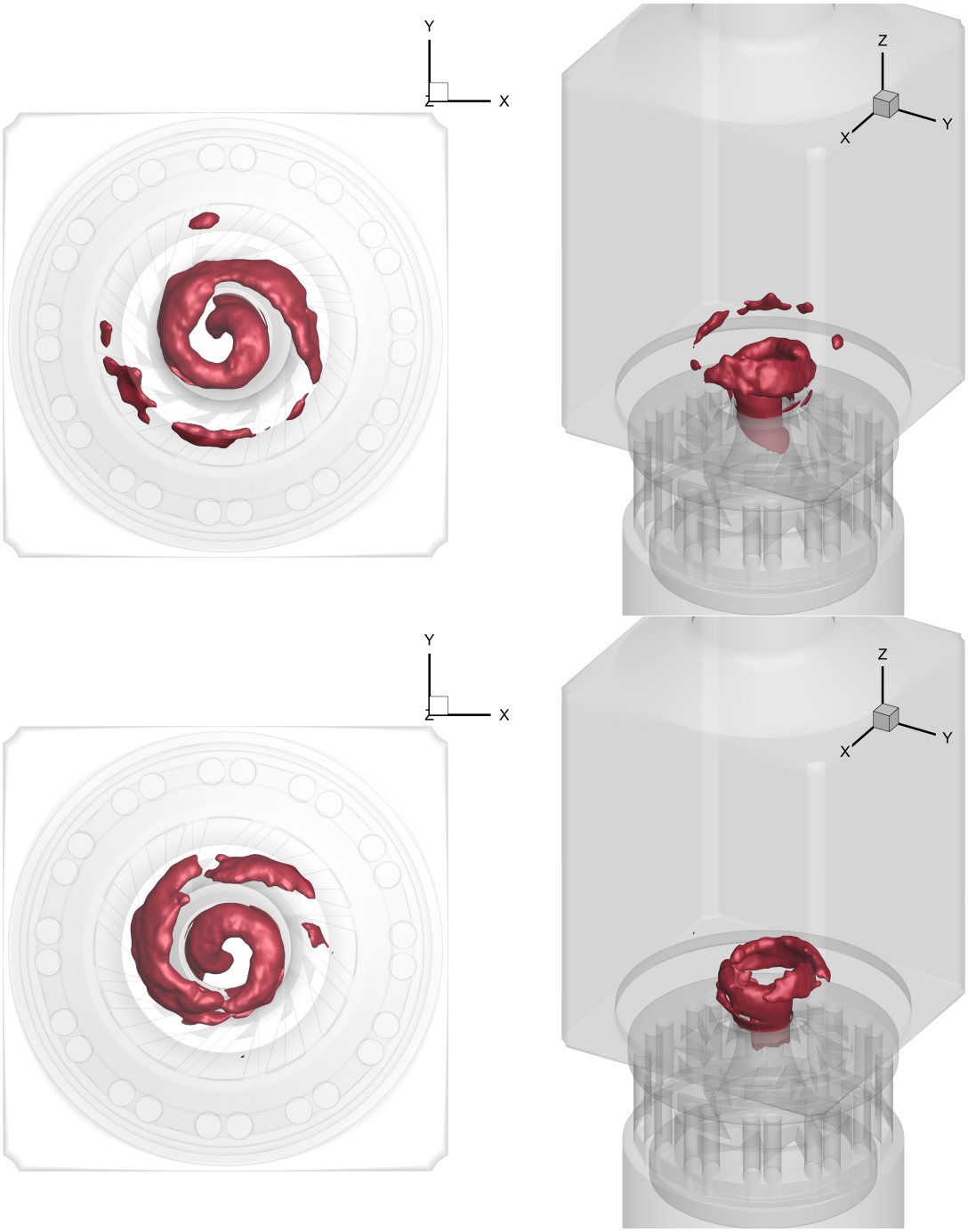


Figure 4.17: 3D DMD modes corresponding to PVC at 0 and .5 of total oscillation period, the isosurface is placed at levels corresponding to 0.2 of the normalized max magnitude.

where $\mathbf{1}$ is a vector of 1s whose outer product with $\overline{\mathbf{q}}$ acts to subtract the $\overline{\mathbf{q}}$ from each column of $\mathbf{Q}$.

Choosing the normalization constants in $\mathbf{P}$ (or $\mathbf{H}$ for primitive variables) such that the variables are scaled to similar orders of magnitude ensures that all variables are considered equally relevant. Choices for computing these normalization constants include the maximum absolute perturbation value and the L2 norm method described in [55].

The trial basis $\mathbf{V}$ is then formed using proper orthogonal decomposition (POD) according to the minimization problem,

$$\mathbf{V} = \underset{\mathbf{A}\in\mathbb{R}^{N\times k};\mathbf{V}\mathbf{V}^T=\mathbf{I}}{\operatorname{argmin}} ||\mathbf{Q}' - \mathbf{A}\mathbf{A}^T\mathbf{Q}'||_2. \tag{4.14}$$

The solution to this problem is typically computed using the singular value decomposition (SVD). When applied to datasets of this scale, the associated memory cost can become extremely large, requiring specialized tools. For example, the basis generation for this relatively small time series requires three matrices to be allocated in memory, each of which is approximately 3 TB in size. To facilitate these prepossessing steps the distributed memory linear algebra tool, PLATFORM, is used to generate these basis sets for ROM usage.

The singular values quantitatively describe the cumulative energy content in the spatial modes. This decay is quantified based on the residual power given by

$$\text{Residual Power } \%(n) = \left(1 - \frac{\sum_{i=1}^{i=n}\sigma_i^2}{\sum\sigma_i^2}\right) \times 100\%. \tag{4.15}$$

However, as seen for the GTMC dataset (Fig. 4.18) there is no clear cut-off, and a significant amount of modes is required to resolve a significant portion of the energy. By computing individual variable PODs we can see that this is primarily driven by the low decay in pressure. Further, decay characteristics are seen to worsen as the training region is extended. Taken together, this presents a significant challenge for ROMs.

4.5.2 Reduced-Order Modeling (ROM) Results

Initially, we use an LSPG-type ROM applied to the Flame A configuration. The static trial basis is generated over a time period of the 5ms described in the previous section consisting of 5000 snapshots from $t = 0.255 - 0.26$s. This corresponds to roughly two flow-through periods of the full computational domain, roughly three acoustic periods, and 10 Precessing Vortex Core (PVC) hydrodynamic cycles. These LSPG ROM runs proved to be unsuccessful as even within the training region they suffered from numerical instability and deviated significantly from the FOM dataset (Fig. 4.20).

The LSPG ROM was unable to provide a stable model even within the training region. Using the same trial bases we then applied the MP-LSVT method. The LSPG and MP-LSVT ROM were both evolved using a 2nd order time implicit time integration scheme consistent with that used for the FOM. The MP-LSVT method improves significantly in terms of stability and is able to fully reconstruct the training region and advance beyond it without suffering a numerical failure. The ROM performance is assessed using unsteady field comparisons at two representative time instances as shown in Fig. 4.19. It can immediately be observed that the overall dynamics within the basis generation region are well-captured. However, regardless of the number of modes used (50 or 90), the prediction shows significant deviation beyond the training region as seen in the smearing and non-physical character of the unsteady field. This deviation is characterized by the flow field "freezing" in place and the fine-scale features begin to smear.

In Fig. 4.20 and Fig. 4.21 the time traces of pressure and temperature are shown. These probes correspond to the locations highlighted in Fig. 4.19. The improvement of the MP-LSVT method over the classical LSPG method is observed within the training region. Even with the relatively small number of modes, the method represents the unsteady dynamics well in the training region, but errors accumulate, and a considerable deviation develops in future state prediction (i.e. beyond the training data). Despite this, the MP-LSVT method shows significant stability improvement over the LSPG as we note the latter method is unable to reconstruct the training region due to numerical

instability.

4.5.3 A priori Projection Error Quantification

While the MP-LSVT offers improved stability, the fundamental limitation is governed by the information lost by projecting on to the reduced basis, referred to as projection error. This shortcoming is immediately apparent in the online ROM runs as they advance beyond the training region The projection error outside the training region varies from problem to problem but can be examined offline to observe the upper limit of any choice of basis projected error. The error of the overall state can be quantified as,

$$\text{Projection Error}(t) = \frac{||\mathbf{q} - \mathbf{V}\mathbf{V}^T\mathbf{q}||_2}{||\mathbf{q}||_2}, \tag{4.16}$$

with the spatial representation given as

$$\text{Field Error}(x,t) = |\mathbf{q} - \mathbf{V}\mathbf{V}^T\mathbf{q}| \tag{4.17}$$

These metrics are applied to the static basis trained over 5000 snapshots representing 5 ms. The results are shown in Fig. 4.22. The projection error within the training region is significantly improved with increased dimension. The improvement is commensurate with the general observations for ROMs that increasing the mode count will monotonically increase the accuracy of the projection. Even with this improvement, outside the training region, the error significantly increases and is insensitive to the variations in the number of modes used. Even with significant increases in the training data, the projected error reaches the same level as even smaller training regions. This can be visualized in the field error (Fig. 4.24) as the maximum error in temperature is less than 100 K. However, when projecting a snapshot beyond the training data, we see large deviations with errors up to 1400 K. This error persists even with significant increases to the training region as seen in Fig. 4.23. This represents a significant limitation of using a linear static basis in a highly chaotic flow with multi-scale advective transport. Increasing the training time will reduce the quality of the projection within the training region for a fixed mode count

as more and more information must be compressed into a series of linear modes.

Ultimately the projection error represents the best possible performance of a projection-based ROM using the corresponding basis. The Kolmogorov N-width [85], describes this worst-case error of a projection onto the best possible static linear subspace. For this flow, the slow decay of the singular values (Fig. 4.18) signifies a slow decay of the Kolmogorov N-width. Two significant limitations are tied to static basis ROMs:

1. To create an adequate projection basis a large FOM dataset is required;

2. The projection outside the training region can be deficient even with an extensive training set.

The large dataset requirement is an issue from two perspectives. The requirement to run a FOM to build a ROM that is only accurate within the training region defeats the overall purpose of ROM development. Even with a very robust training dataset, there is no guarantee that the projection basis will be adequate for a variety of conditions. The proposed solution to mitigate both of these concerns is the adaptive basis method.

4.5.4 Adaptive Basis

While the MP-LSVT method improves the robustness and accuracy of the ROM within the training region, the predictive capabilities (e.g., future-state prediction) are still restricted mainly by the quality of the basis projection, as demonstrated in the previous section.

Adaptive-basis methods (e.g. [86]) have the potential to address the challenge of the slow decay of the Kolmogorov N-width. These methods aim to modify the linear basis online to mitigate the projection error,

$$\mathbf{V}_p^n \triangleq \underset{\mathbf{V}^n \in \mathbb{R}^{M\times k}, \tilde{\mathbf{q}}_p \in \mathbb{R}^{\text{RangeV}}}{\operatorname{argmin}} \|\mathbf{Pr}(\tilde{\mathbf{q}}_p^n)\|_2^2 \tag{4.18}$$

where $\mathbf{r}$ is the fully-discrete FOM equation residual defined in Eq. 2.18, $\tilde{\mathbf{q}}_p^n = \overline{\mathbf{q}}_p + \mathbf{H}^{-1}\mathbf{V}_p^n\hat{\mathbf{q}}_p^n$, and $\tilde{\mathbf{q}}_p^{n-j} = \overline{\mathbf{q}}_p + \mathbf{H}^{-1}\mathbf{V}_p^{n-j}\hat{\mathbf{q}}_p^{n-j}$ This minimization problem is solved exactly

via the update

$$\mathbf{V}_p^n = \mathbf{V}_p^{n-1} + \boldsymbol{\delta} \tag{4.19}$$

where the basis at time-step $n-1$ is adapted to n through an increment, $\boldsymbol{\delta} \in \mathbb{R}^{M\times k}$ given by

$$\boldsymbol{\delta} = \frac{[\mathbf{q}_p^n - \tilde{\mathbf{q}}_p^n](\hat{\mathbf{q}}^n)^T}{\|\hat{\mathbf{q}}^n\|_2^2} \tag{4.20}$$

where $\mathbf{q}^n \in \mathbb{R}^M$ represents the re-projected full-state information, which is evaluated based on the FOM residual explicitly as

$$\begin{aligned}\mathbf{q}(\mathbf{q}_p^n) + \sum_{j=1}^{l} \alpha_j \mathbf{q}(\tilde{\mathbf{q}}_p^{n-j}) - \Delta t \beta_0 \mathbf{f}(\tilde{\mathbf{q}}_p^n, t^n) - \\ \Delta t \sum_{j=1}^{l} \beta_j \mathbf{f}(\tilde{\mathbf{q}}_p^{n-j}, t^{n-j}) = 0,\end{aligned} \tag{4.21}$$

or implicitly

$$\begin{aligned}\mathbf{q}(\mathbf{q}_p^n) + \sum_{j=1}^{l} \alpha_j \mathbf{q}(\tilde{\mathbf{q}}_p^{n-j}) - \Delta t \beta_0 \mathbf{f}(\mathbf{q}_p^n, t^n) - \\ \Delta t \sum_{j=1}^{l} \beta_j \mathbf{f}(\tilde{\mathbf{q}}_p^{n-j}, t^{n-j}) = 0,\end{aligned} \tag{4.22}$$

Here the application of $\mathbf{q}(\mathbf{q}_p)$ is the conversion of primitive variable state to conservative state.

It should be noted that we adopt an alternate, and simplified formulation compared to Peherstofer [86] by updating the basis based on the full-state information $\mathbf{q}$ evaluated at the current time step, n, instead of collecting multiple time steps. We refer to this formulation as the one-step adaptive-basis approach. The method of [86] requires several matrix operations (e.g. SVD, pseudo-inverse) on the re-projected full-state information (i.e. $\mathbf{q}^n$ in Eq. 4.20) collected at multiple time steps during the ROM calculation. Though these matrix operations are trivial for small-scale test problems (e.g. 1D or small-scale 2D), they become challenging regarding implementation for the target large-scale 3D simulation in this work, especially on the CPU memory requirements to store the information necessary for these matrix operations. Additionally, the simplified formula

presents a straightforward way to achieve basis adaptation without worrying about the implementation for large-scale problems.

We now apply the methodology to the GTMC dataset. For the previous ROM cases the predictive region begins at $t = 0.26$ s. An initial 20 mode basis, which will be adapted through the run, is generated using the 20 snapshots of the FOM data $t = 0.2599980 - 0.26$ s. From a ROM pre-processing perspective, this is a significant reduction in training as the static basis method leveraged a set of 5,000 snapshots spaced from $t = 0.255 - 0.26$ s. The adaptive basis ROM is then run over the same predictive region as the previous static basis ROMs for one millisecond corresponding to 2 PVC cycles. The basis is adapted at every time step beyond the training region.

Comparisons of temperature and pressure traces are shown in Fig. 4.25 focused in the predictive region. As noted previously, the static basis ROM degrades immediately upon exiting the training region. In contrast, the adaptive basis ROM follows the low-frequency behavior and replicates the high-frequency oscillation in both pressure and temperature.

We output two representative fields Fig. 4.26 at time instances noted in Fig. 4.25. The coherent structures of the swirling flow show significant improvement with none of the smearing behavior seen in the static basis method, representing a significant improvement over the static basis. We additionally compare the field error using the relationship

$$\% \text{ error} = \frac{\mathbf{q}_{FOM} - \mathbf{q}_{ROM}}{\mathbf{q}_{FOM}} \times 100\%. \tag{4.23}$$

It should be noted that while the instantaneous fields are contours of temperature, the error fields are shown for the full state $\mathbf{q}$. We note the error is limited to regions near the gradients of the flame. However, the maximum error does not grow significantly with both time steps shown having a maximum magnitude of approximately 18%. This error behaviour is expected to continue with error being limited to areas of local gradients. This results in an offset of local mixing layer location while maintaining overall dynamic evolution.

Overall, the one-step adaptive basis method overcomes both the Kolmogorov N-width problem and significantly reduces the FOM data required. From an application perspec-

tive, the FOM run time is trivial (20 time-steps in this case), followed by switching to the adaptive ROM. This speaks to greater practicality compared with static ROM, for which, depending on the problem, significant resources must be spent on the FOM data collection.

Adaptive basis significantly improves the predictive capability of the ROM as well as the operational characteristics. The robustness of the MP-LSVT method combined with the basis adaptation has shown good agreement with the full-order model. However, the performance is enabled by the evaluation of the full-order model at every basis update step. In fact, from a computational cost perspective, the adaptive basis ROM is more expensive than if an equivalent FOM was conducted. This raises the question of what has been achieved. The final piece that is needed to make these ROMs actually efficient compared with their constituent FOMs is sampling-based hyper reduction methods. These sampling methods and integration into the larger ROM framework will be discussed in the following chapter. Even without the sampling the GTMC ROM is among the largest projection-based ROMs conducted and has required significant method and operational improvements to run.

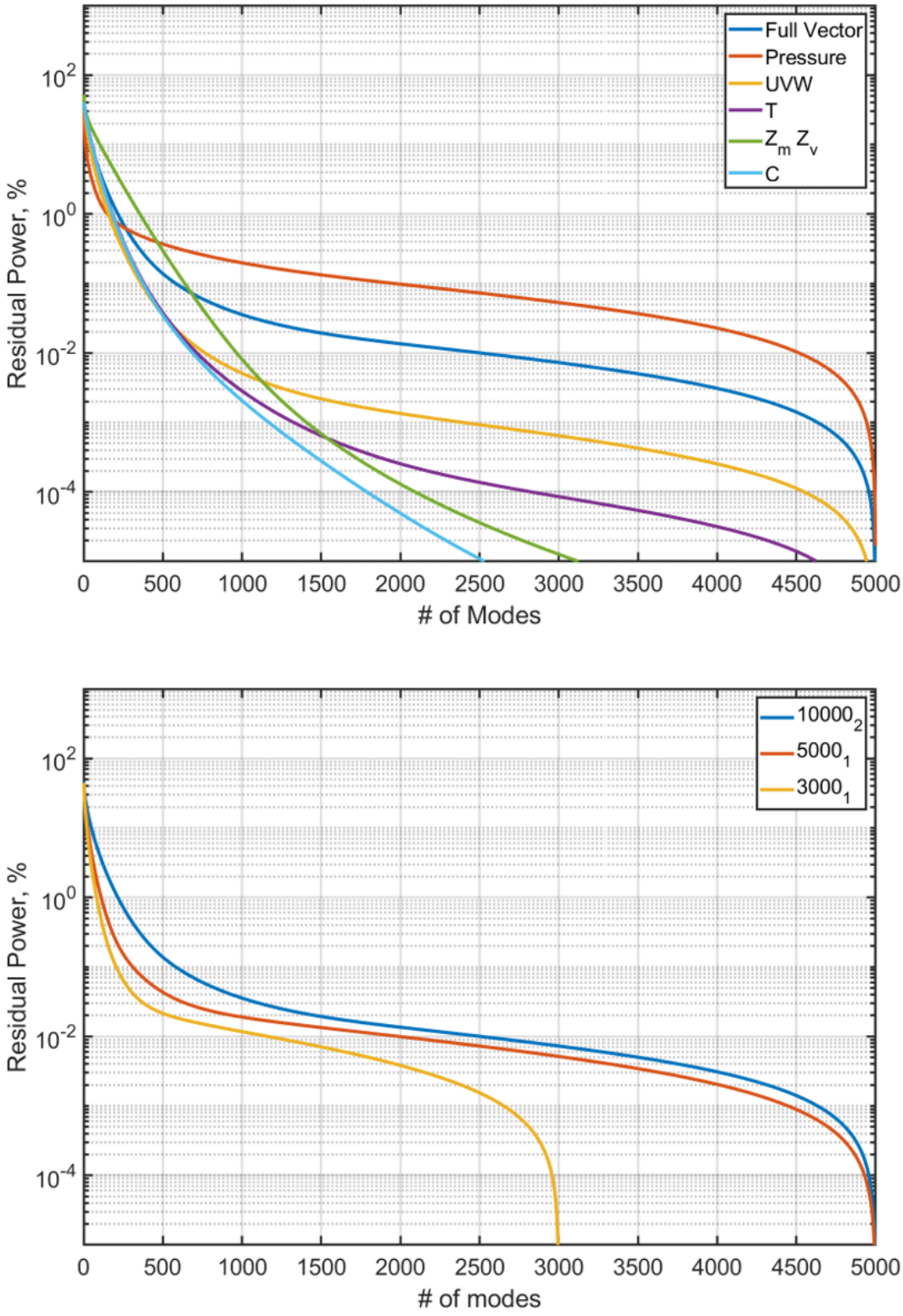


Figure 4.18: Singular value decay residual for GTMC for various variable groupings for a 5000 snapshot training region (top) and training region lengths (bottom).

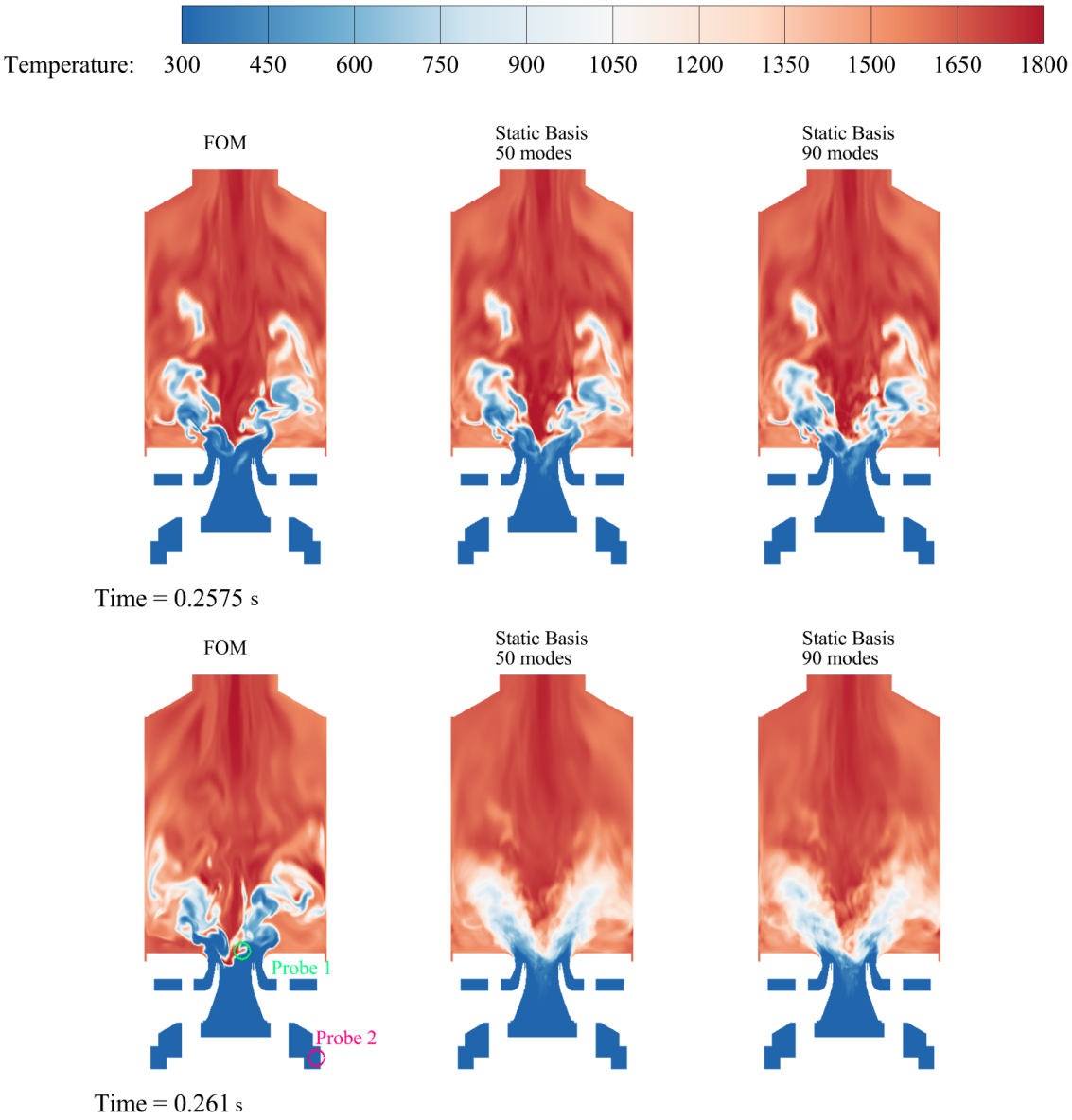


Figure 4.19: Instantaneous online ROM fields at 0.5 (Top) and 1.2 (Bottom) of total training time with Fig. 4.20and 4.21 locations highlighted. Training Window: $t = 0.255 - 0.26$s.

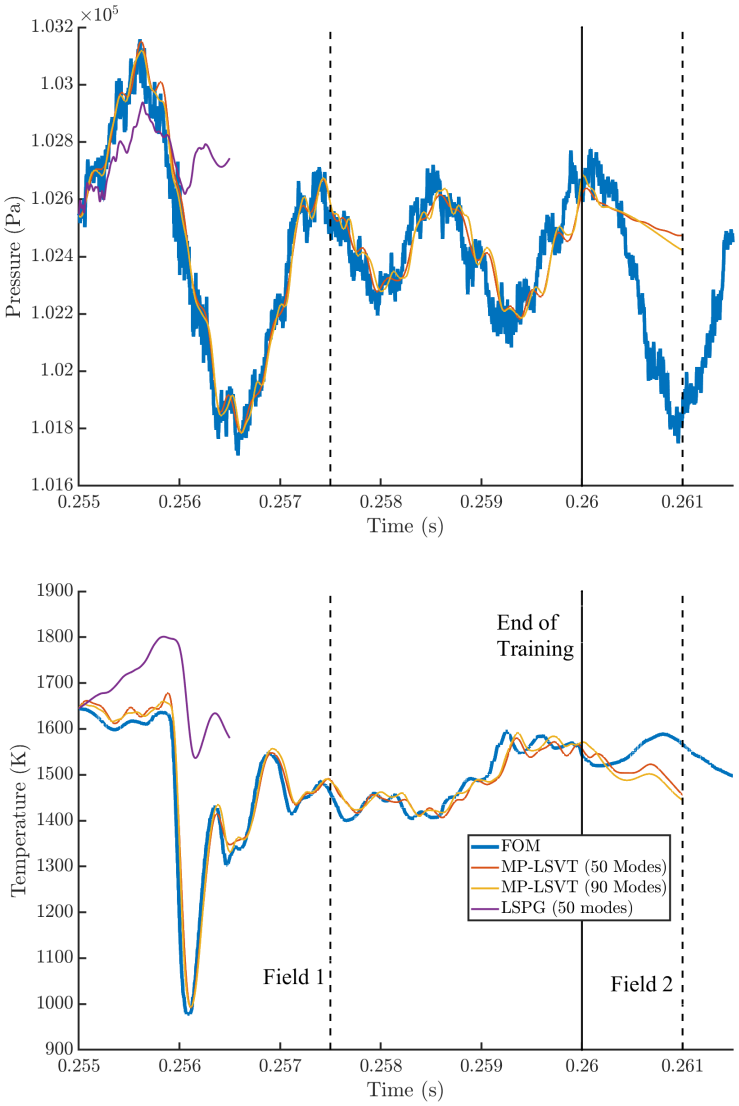


Figure 4.20: Pressure and temperature probes within flame front for various static basis choices. Training Window: $t = 0.255 - 0.26$s. Probe location is probe 1 as visualized in Fig. 4.19.

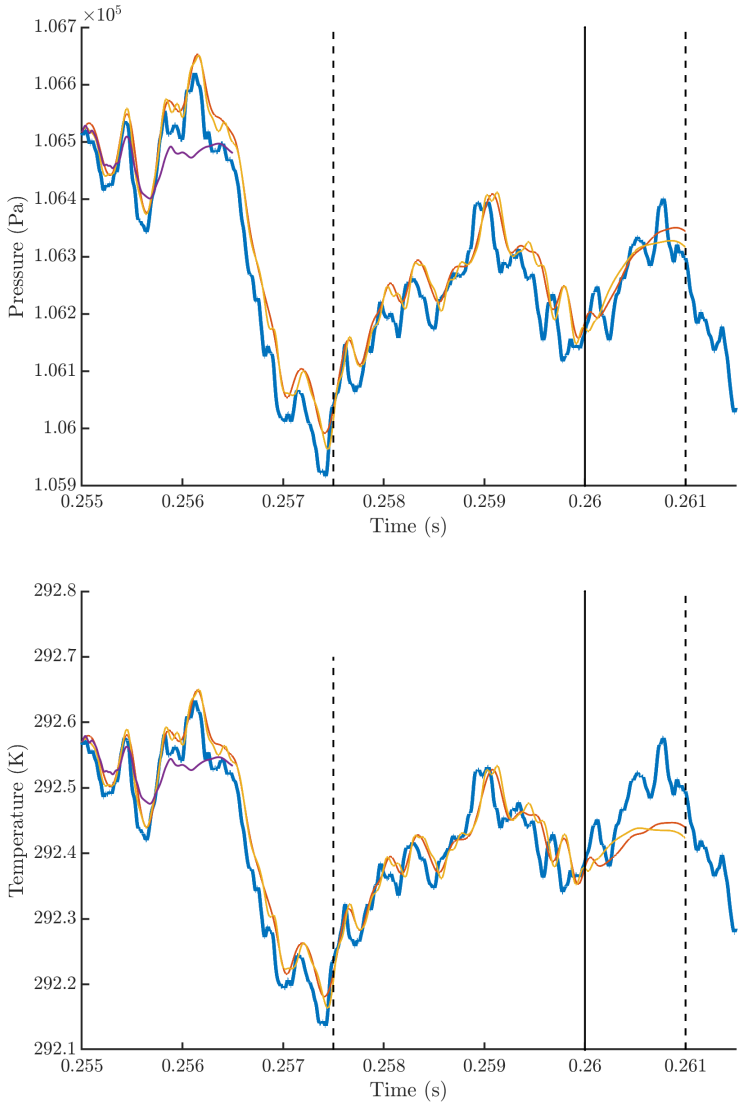


Figure 4.21: Pressure and temperature probes within flame front(Top) and plenum(Bottom) for various static basis choices. Training Window: $t = 0.255 - 0.26$s. Probe location is probe 2 as visualized in Fig. 4.19.

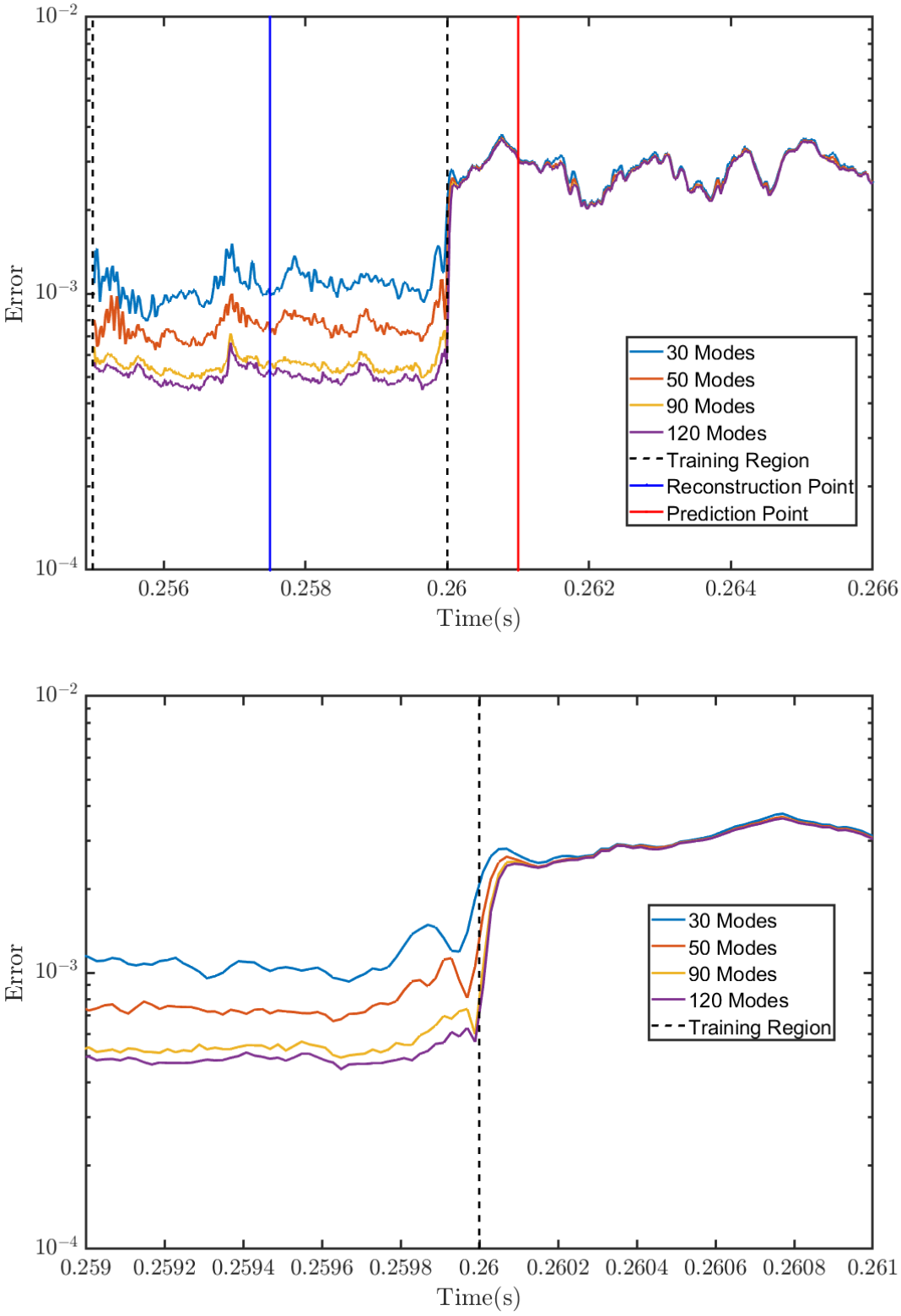


Figure 4.22: Static basis projection error for various mode counts for 5 ms training window (left) and zoomed close to end of training (right).

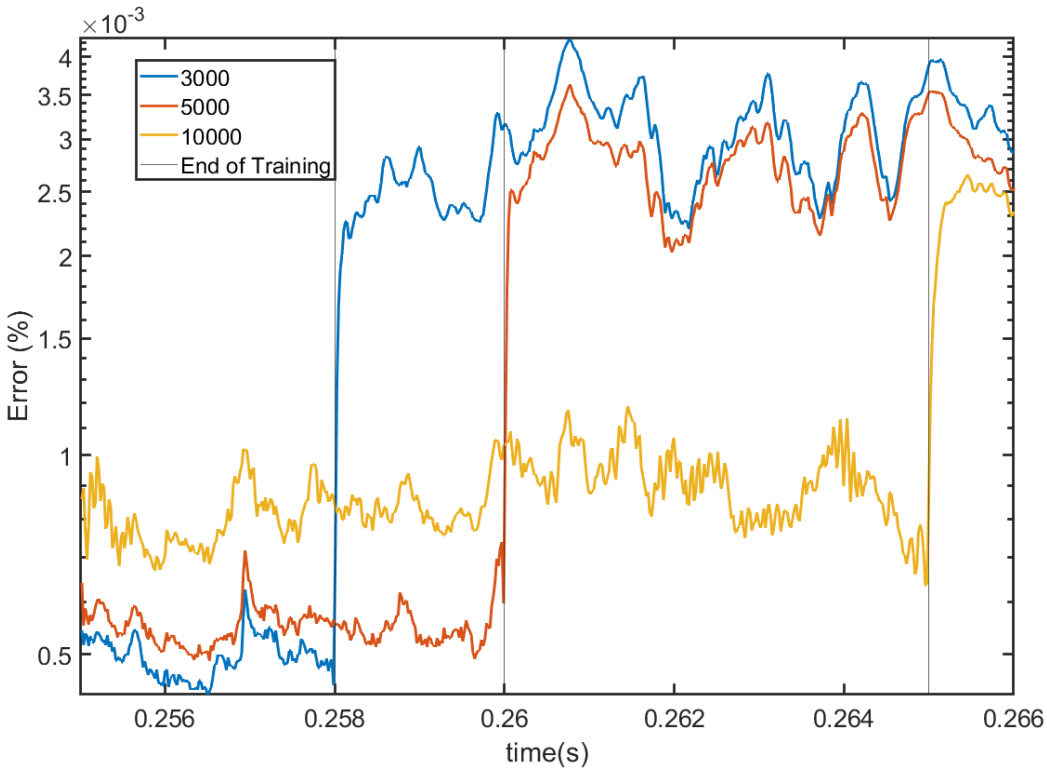


Figure 4.23: Static basis projection error for 90 modes computed for various training lengths.

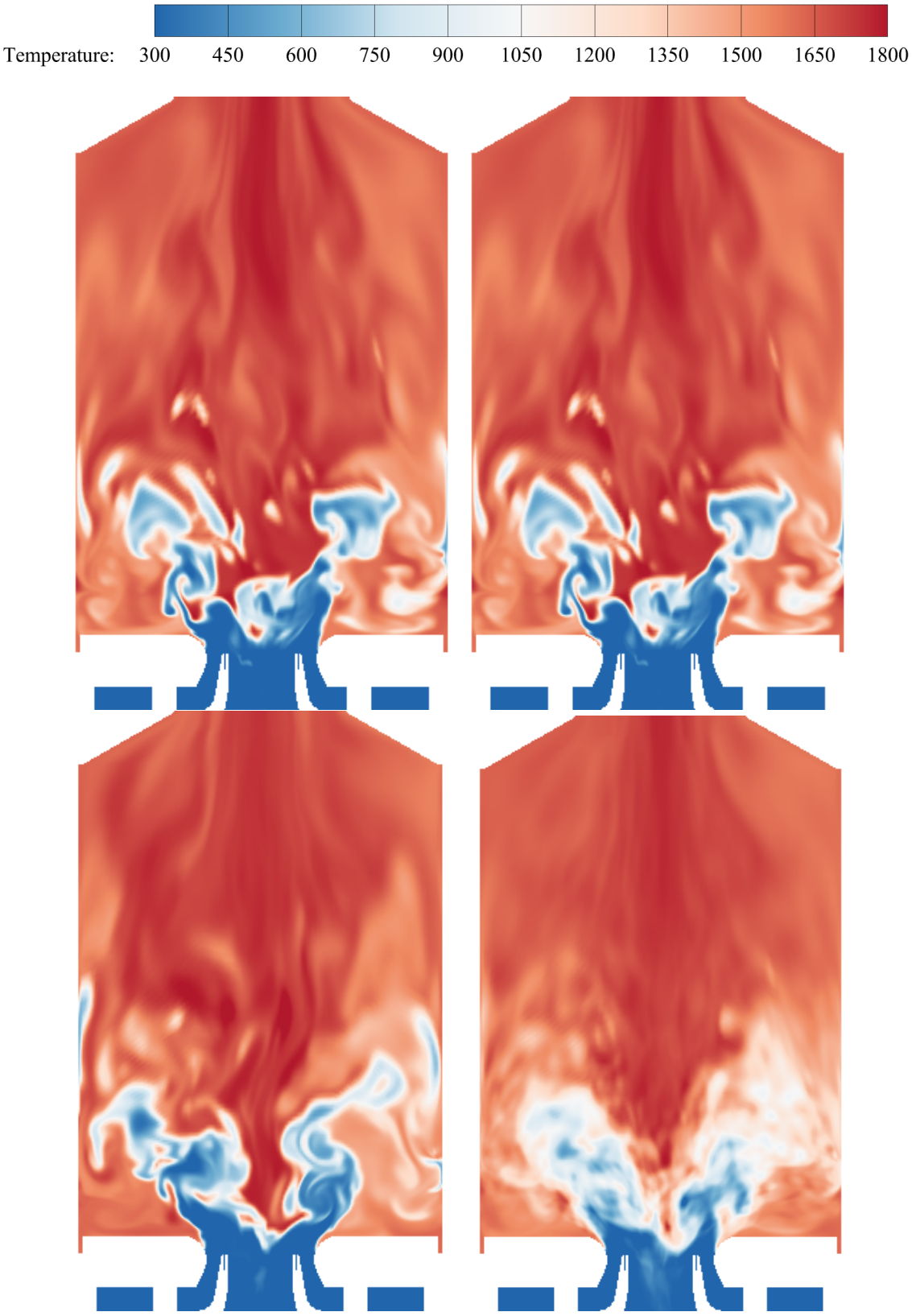


Figure 4.24: Full-order model (Left) temperature field compared with a priori static basis 90 mode projection (Right) at $t = .2575$s (Top) and $t = .261$s (Bottom) These snapshots correspond to the identified time instances in Fig. 4.20 .

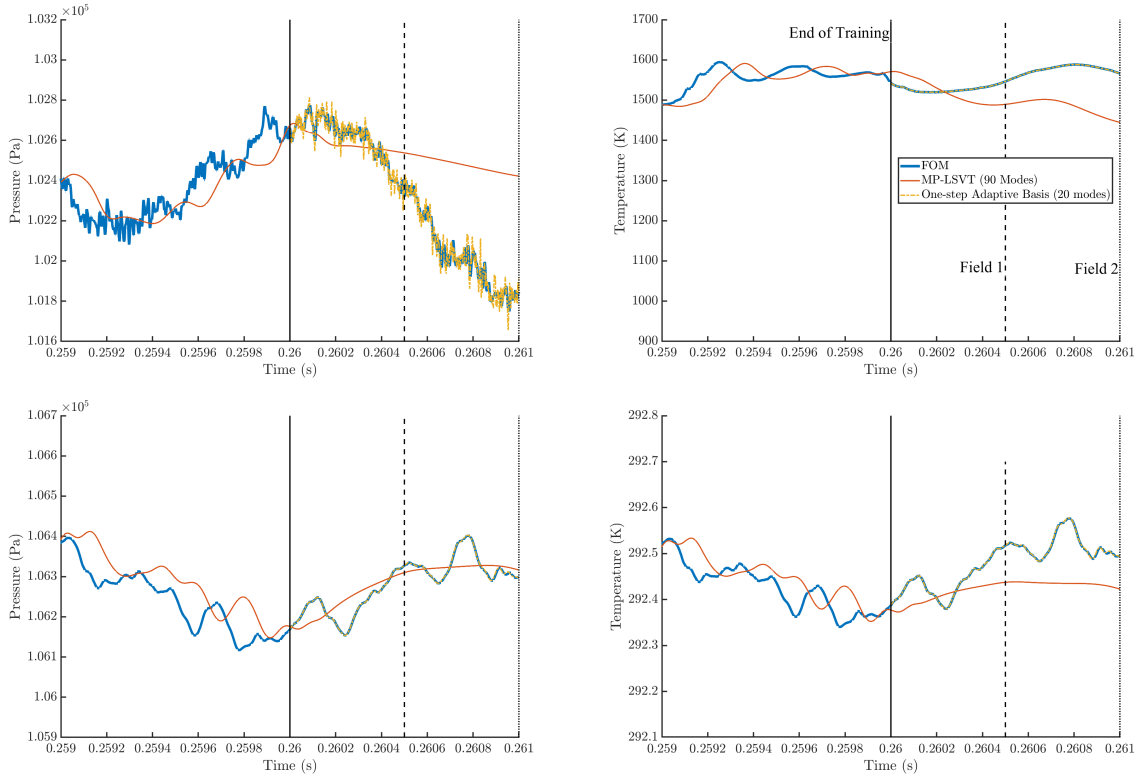


Figure 4.25: Pressure and temperature probes within the flame front (Top) and plenum (Bottom) for static and adaptive basis reduced-order models compared with the full-order model focused on the predictive region. Note that the training region of the static basis extends for $t = 0.255 - 0.26$ s.

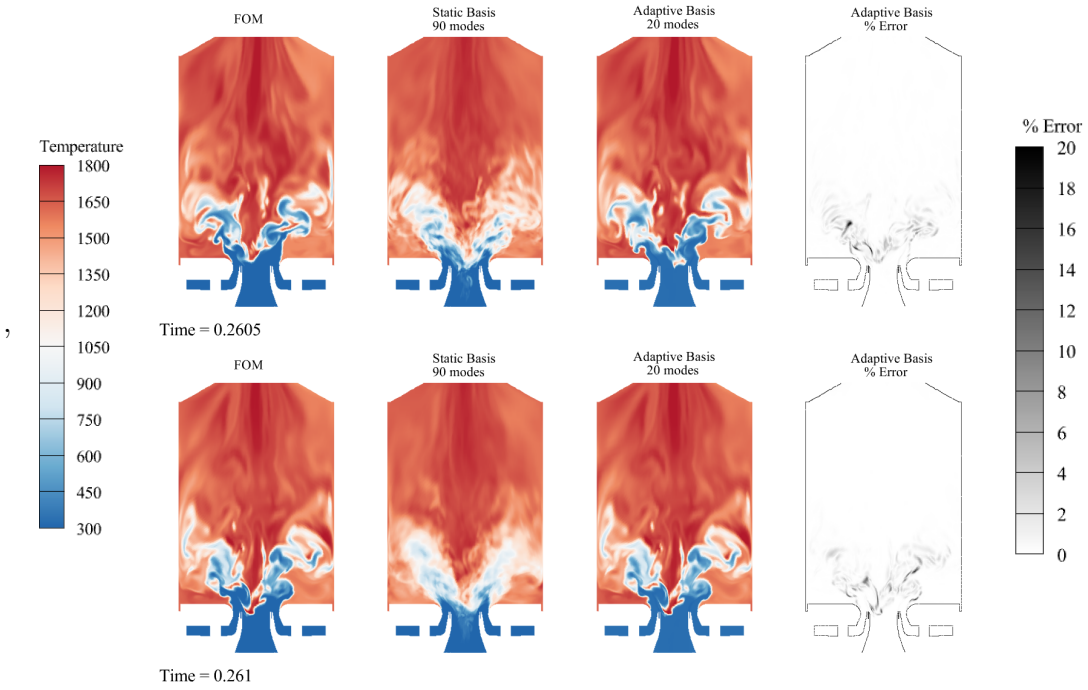


Figure 4.26: Instantaneous online temperature fields for FOM and static and adaptive ROMs with relative field error for the adaptive ROM.

Chapter 5

Adaptive Sampling: Predictive Capabilities

5.1 Outline

The previous chapter applied projection ROMs to problems of unexplored scale and complexity. Unique computing challenges relevant to problem size were overcome, stability was improved via the application of the MP-LSVT method, and predictive accuracy was enabled with an adaptive basis method. The synthesis of these methods combined with the scale and complexity of the problem represents another step in the application of ROM methods to real problems. For practical use, however, it is obvious that the missing piece, meaningful computational speed-up, is required

As discussed in the GTMC chapter, evaluating both the static and adaptive ROM requires the evaluation of the non-linear function, which begs the question of what is achieved by using the projection-based ROM. Until now, the discussions and results shown for ROM-type methods do not generate any computational savings. Evaluating the non-linear term $\mathbf{f}$ requires the full dimensional state. This is not a problem in projection-based ROM development steps on linear time-invariant problems, as the operator defining the system's evolution can be preconditioned independently of the problem. Due to the non-linear operator being fundamentally dependent on the current state, very little can be done to avoid it.

This chapter will focus on the sampling required to achieve meaningful speedup and consists of three parts.

- First, a description of the traditional usage of sampling methods for projection-

based ROMs is provided. The adaptive sampling method, which integrates the general sampling method together with the adaptive basis method, is introduced;

- The adaptive sampling method's capabilities will be presented on two sample problem types focusing on predictive accuracy and computational efficiency;
- Finally, some applications of the method to enhance existing simulation frameworks are shown. The methods shown are meant to advocate for possible ROM use cases in larger computational frameworks where a pure ROM may be undesirable.

This chapter will focus primarily on the method and accuracy of various applications with the following chapter aimed at looking at the computational challenges of integrating this method with a computational solver.

5.2 Review of Hyper-Reduction Methods

As a starting point, recall the traditional hyper-reduction methods introduced in chapter 2. Fundamentally these are derived to overcome the "lifting" problem of the standard ROM equation,

$$\mathbf{W}^T\mathbf{V}\frac{\mathrm{d}\hat{\mathbf{q}}}{\mathrm{d}t} = \mathbf{W}^T\mathbf{P}\mathbf{f}(\tilde{\mathbf{q}}, t). \tag{5.1}$$

Even if a significant dimensional reduction is achieved via compression of the full state $\mathbf{q}$ to the reduced state $\hat{\mathbf{q}}$ the full state must still be used to evaluate $\mathbf{f}$ at every state spatial point. Given the evaluation of the component flux and source, terms dominate a fluid flow solver; if the nonlinear term must always be evaluated at the full state, no speed-up is usually be achieved.

Hyper-reduction methods eliminate this requirement by estimating the nonlinear function via a small subset of evaluated points. Mathematically this is represented via a sampling matrix $\mathbf{S}$ wich selects degrees of freedom to evaluate. The ROM equation describing the evolution of the reduced states for a statically hyper-reduced ROM is given by

$$\tilde{\mathbf{q}}_p^n \triangleq \underset{\tilde{\mathbf{q}}_p \in \text{Range}(\mathbf{V}_p)}{\text{argmin}} ||\mathbf{U}[\mathbf{S}^T\mathbf{U}]^+\mathbf{S}^T\mathbf{Pr}(\tilde{\mathbf{q}}_p^n)||_2^2 \tag{5.2}$$

with a corresponding test basis of

$$\mathbf{W}_p^n \frac{\partial \mathbf{U}[\mathbf{S}^T\mathbf{U}]^+\mathbf{S}^T\mathbf{Pr}(\tilde{\mathbf{q}}_p^n)}{\partial \hat{\mathbf{q}}^n} = \mathbf{U}(\mathbf{S}^T\mathbf{U})^+\mathbf{S}^T \frac{\partial \mathbf{Pr}(\tilde{\mathbf{q}}_p^n)}{\partial \hat{\mathbf{q}}^n}. \tag{5.3}$$

The ROM takes the form,

$$[\mathbf{S}^T\mathbf{W}_p^n]^T \left[[\mathbf{S}^T\mathbf{V}]^+\right]^T [\mathbf{S}^T\mathbf{V}]^+\mathbf{S}^T\mathbf{Pr}(\tilde{\mathbf{q}}) = 0, \tag{5.4}$$

where the approximate state is given by $\tilde{\mathbf{q}} = \mathbf{q}_{\text{center}} + \mathbf{H}\mathbf{V}_p^{n-1}\hat{\mathbf{q}}$. Recall that $[]_p$ refers to the primitive form of the vector [] and $\mathbf{H}$ is the corresponding normalization matrix for primitive variables.

Static-sampled ROMs have seen remarkable success and have run up to 1000 times faster with as few as 0.1% of the total mesh elements [52]. Additional computational practicalities are also enabled in that the full state dimension can be avoided entirely during the online computation portion. A simulation that might have required a cluster scale system could be performed on a personal scale computer. Unfortunately, while highly efficient, these methods have two significant drawbacks.

1. Regardless of the sampling method used, if the system dynamics diverge significantly from the training region the sampling points chosen will be insufficient. This type of deviation is very likely to occur in studies that vary the operating conditions of the system;

2. The capability of the ROM to predict either in a predictive or parametric setting is still limited by the linear static basis used in the offline step to generate the ROM. Using a basis adaptation method with static sampling is not advisable, as the sampled points are inherently tied to the training data.

Thus, the core concept of sampling-based methods is required to achieve computational efficiency of ROMs; Any proposed solution must work together with the adaptive

basis method described in the previous chapter and adapt sampling points as the problem dynamics change dynamically. The computational framework used to achieve this is referred to as adaptive sampling.

5.3 Adaptive Sampling Method

5.3.1 One-step Adaptive Basis method

An adaptive DEIM method was first proposed by Peherstorfer [87, 86]. This method introduced the idea of using successive low-rank updates to adapt the trial basis $\mathbf{V}$. The basis update was formulated to based on previous full-order model snapshots. More recently, Huang et al. [88] proposed a simplified formulation compared to Peherstorfer. Synthesizing the optimization problems from the adaptive basis (Eqn. 4.18) and hyper-reduction (Eqn. 2.23) methods, a combined method would ideally optimize both the sampling points and basis.

$$\{\tilde{\mathbf{q}}^n, \mathbf{V}_p^n, \mathbf{S}^n\} \triangleq \underset{\tilde{\mathbf{q}}\in\mathbb{R}^k, \mathbf{V}_p^n\in\mathbb{R}^{M\times k}, \mathbf{S}^n\in\mathbb{R}^{M\times s}}{\operatorname{argmin}} ||\mathbf{U}[(\mathbf{S}^n)^T\mathbf{U}]^+(\mathbf{S})^T\mathbf{P}\mathbf{f}(\mathbf{q}_p)||_2^2 \tag{5.5}$$

The sampling matrix $\mathbf{S} \in \mathbb{R}^{M\times s}$ is a section matrix that selects the degrees of freedom of the full state used in the hyper-reduced ROM. The latent state $\hat{\mathbf{q}}$ and trial basis $\mathbf{V}$ are likewise updated.

This least-squares problem form proves to be prohibitively expensive to solve directly. Recall that any solution here must be repeated at every sampling update step. If the sampling update is more expensive than the final sampled ROM, the method will be too impractical. With this limitation, the basis and sampling points are instead updated independently. This approach updates the reduced state $\hat{\mathbf{q}}$, the projection trial basis $\mathbf{V}$, and the selected sampling points $\mathbf{S}$ sequentially.

Basis Update

The adaptation of the basis $\mathbf{V}$

$$\mathbf{V}_p^n = \mathbf{V}_p^{n-1} + \boldsymbol{\delta}, \tag{5.6}$$

which is updated using the same method described in the adaptive basis method for the GTMC

$$\boldsymbol{\delta} = \frac{[\mathbf{q}_p^n - \tilde{\mathbf{q}}_p^n](\hat{\mathbf{q}}^n)^T}{\|\hat{\mathbf{q}}^n\|_2^2}, \tag{5.7}$$

where $\mathbf{q} \in \mathbb{R}^M$ is the full state computed from the FOM residual. In the GTMC results presented earlier, this adaptation occurs at every mesh point and, as a result, does not provide computational acceleration. Thus the adaptive ROM is formulated to update the basis at the selected sampling points during most iterations.

$$\mathbf{S}_{n-1}^T \mathbf{V}_p^n = \mathbf{S}_{n-1}^T \mathbf{V}_p^{n-1} + \mathbf{S}_{n-1}^T \boldsymbol{\delta} \tag{5.8}$$

State Update

Combining this one-step update with the sampled ROM (Eqn. 5.4) we use the basis update of the previous timestep to form the adaptive sampled ROM

$$[\mathbf{S}_{n-1}^T \mathbf{W}_p^n]^T \left[[\mathbf{S}_{n-1}^T \mathbf{V}_{n-1}]^+\right]^T [\mathbf{S}_{n-1}^T \mathbf{V}_{n-1}]^+ \mathbf{S}_{n-1}^T \mathbf{Pr}(\tilde{\mathbf{q}}^n) = 0, \tag{5.9}$$

Where the evolution of the ROM to timestep n is defined by the updated sampling and trial basis of the previous time step.

Coherence is a critical aspect discussed previously in the application of projection-type ROMs to fluid flow problems. Problems that contain significant chaos, like reacting flows, are challenging. From the perspective of representing the full state as a set of linear vectors, it is immediately apparent that a highly chaotic flow significantly limits the possible data compression. Additionally, hyper-reduction regions of rich local features (such as shocks or flame fronts) are challenging as they may change the ideal sampling locations. Huang et al. [88] proposed a method that incorporates non-local information into the full-state estimates, which improves the accuracy of these methods for multi-scale problems.

Thus we have two types of iterations. The first operates with a timestep Δt where the residual is evaluated only at the sampled points, and the latent and sampled states are updated together via,

$$\begin{aligned}
&\mathbf{S}_{n-1}^T\mathbf{q}(\mathbf{S}_{n-1}\overbrace{\mathbf{S}_{n-1}^T\mathbf{q}_p^n}^{\text{sampled state}}+\mathbf{S^*}_{n-1}\mathbf{S^*}_{n-1}^T\tilde{\mathbf{q}}_p^n,t^n)+\sum_{j=1}^{l}\alpha_j\mathbf{S}_{n-1}^T\mathbf{q}(\tilde{\mathbf{q}}_p^{n-j})\\
&-\Delta t\beta_0\mathbf{S}_{n-1}^T\mathbf{f}(\mathbf{S}_{n-1}\overbrace{\mathbf{S}_{n-1}^T\mathbf{q}_p^n}^{\text{sampled state}}+\mathbf{S^*}_{n-1}\mathbf{S^*}_{n-1}^T\tilde{\mathbf{q}}_p^n,t^n)-\Delta t\sum_{j=1}^{l}\beta_j\mathbf{S}_{n-1}^T\mathbf{f}(\tilde{\mathbf{q}}_p^{n-j},t^{n-j})=0.
\end{aligned}\tag{5.10}$$

Where the unsampled cell selection matrix is given by $\mathbf{S}^* \in \mathbb{R}^{M\times M-s}$ composed of the points not contained in $\mathbf{S}$. The second iteration operates with a timestep of $z_s\Delta t$ which requires the evaluation of the full dimensional residual at every mesh point. This full-dimensional basis update will coincide with the sampling update. The hyper-parameter of z_s directly defines the efficiency of the ROM as we aspire to evaluate the full dimensional ROM as infrequently as possible. The formulation of this long and short timestep is inspired by the scale separation of dynamical systems with high-frequency coherency (hydrodynamic and flame features) contrasting with low-frequency coherent features (low-frequency acoustics like those found in the GTMC).

$$\begin{aligned}
&\mathbf{S}_{n-1}^T\mathbf{q}(\mathbf{S}_{n-1}\overbrace{\mathbf{S}_{n-1}^T\mathbf{q}_p^n}^{\text{sampled state}}+\mathbf{S^*}_{n-1}\overbrace{\mathbf{S^*}_{n-1}^T\mathbf{q}_p^n}^{\text{unsampled state}},t^n)+\sum_{j=1}^{l}\alpha_j\mathbf{S}_{n-1}^T\mathbf{q}(\tilde{\mathbf{q}}_p^{n-jz_s})\\
&-\Delta tz_s\beta_0\mathbf{S}_{n-1}^T\mathbf{f}(\mathbf{S}_{n-1}\overbrace{\mathbf{S}_{n-1}^T\mathbf{q}_p^n}^{\text{sampled state}}+\mathbf{S^*}_{n-1}\overbrace{\mathbf{S^*}_{n-1}^T\mathbf{q}_p^n}^{\text{unsampled state}},t^n)-\Delta tz_s\sum_{j=1}^{l}\beta_j\mathbf{S^*}_{n-jz_s}^T\mathbf{f}(\tilde{\mathbf{q}}_p^{n-jz_s},t^{n-jz_s})=0.
\end{aligned}\tag{5.11}$$

A schematic of how these time-steps works is provided in Fig 5.1. It is apparent how the hyper-parameter z_s interacts with the ROM method. As z_s is increased, the number of sampled iterations (Eqn. 5.10) relative to unsampled iterations (Eqn. 5.11) increases thereby diminishing the number of times the full dimension non-linear function must evaluate.

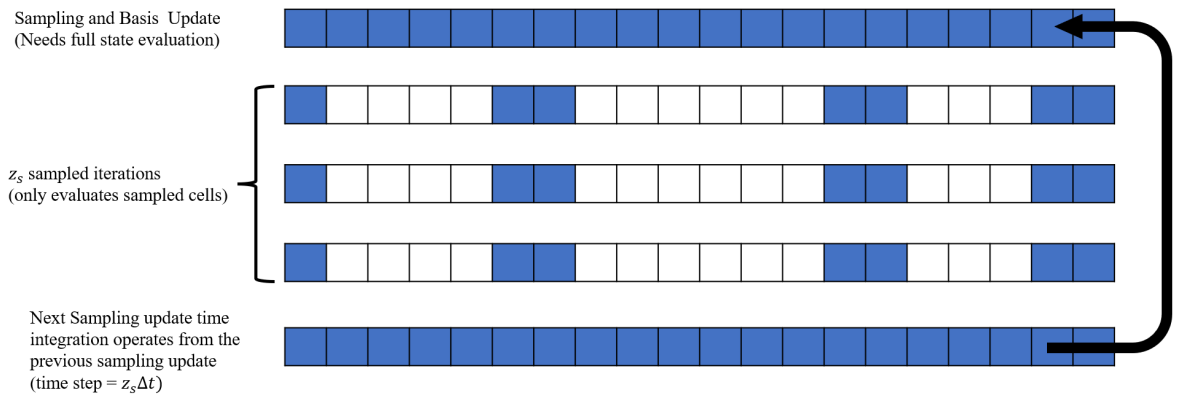


Figure 5.1: Visualization of the non-local time stepping.

Sampling Update

The sampling points $\mathbf{S}$ are updated via the minimization problem,

$$\mathbf{S}_n \triangleq \underset{\mathbf{S}_n \in \mathbb{R}^{M\times s}}{\operatorname{argmin}} ||\mathbf{q}_p^n - \mathbf{V}_p^n(\mathbf{S}_n^T\mathbf{V}_p^n)^+\mathbf{S}_n^T\hat{\mathbf{q}}_p^n||_2^2. \tag{5.12}$$

Practically this is achieved by calculating the interpolation error $\mathbf{e} = \mathbf{q}_p^n - \mathbf{V}_p^n(\mathbf{S}_n^T\mathbf{V}_p^n)^+\mathbf{S}_n^T\hat{\mathbf{q}}_p^n$. The error at each point is sorted by magnitude, and the first s points are chosen as the new set sampling points. This point selection is quasi-optimal with regard to the upper bound of the adaptation error. In practice, both this update and the basis adaptation require evaluation of the full dimension residual. As a result, for all practical cases, we adapt the basis and sampling points at the same time.

5.3.2 Linear Solver

To better illustrate how this ROM evaluation differs from a classical FOM we start with a description of the GEMS solver when operating as a FOM solver. This is illustrated in algorithm 3. In a finite volume solver, the state vector (pressure, velocity, temperature, etc.) is collocated at each volume element. The evolution of the system can be defined as

$$\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} = \mathbf{f}(\mathbf{q}), \tag{5.13}$$

The non-linear function describes how the system $\mathbf{q}$ evolves, which is defined by the governing equations from chapter 2. For the full-order model, the evolution of each element of $\mathbf{q}$ is defined by its local and surrounding states

This leads to the formation of a sparse linear system which describes the evolution of M degree of freedom system. An iterative method is usually used to time march a sparse system based on a non-linear function. The solution of the non-linear system is approximated using a Newton method where $\mathbf{J}(\mathbf{q}) = \frac{\partial \mathbf{f}(\mathbf{q})}{\partial \mathbf{q}} \in \mathbb{R}^{M \times M}$ is the Jacobian of the nonlinear function $\mathbf{f}$ evaluated around the state $\mathbf{q}$.

$$\mathbf{f}(\mathbf{q} + \Delta\mathbf{q}) \approx \mathbf{f}(\mathbf{q}) + \mathbf{J}(\mathbf{q})\Delta\mathbf{q} \tag{5.14}$$

from an initial $\mathbf{q}$ the advancement of the state $\Delta\mathbf{q}$ is solved via a linear system.

$$\mathbf{J}(\mathbf{q})\Delta\mathbf{q} = -\mathbf{f}(\mathbf{q}) \tag{5.15}$$

The solved $\Delta\mathbf{q}$ is used to update the current state via

$$\mathbf{q} = \mathbf{q} + \Delta\mathbf{q}. \tag{5.16}$$

Eqn. 5.15 and Eqn. 5.16 are iterated until the system is converged with the final state representing the advanced systems $\mathbf{q}^{n+1}$. The previous timesteps are then updated, the solution is advanced, and the next time-step computation begins. The overall structure of the solver is described in algorithm 3.

With this overall solver structure in mind, we examine how adaptive sampling changes this process. In the context of a FOM solver, each iteration and sub-iteration type is functionally identical to the previous as far as the execution path. Every residual calculation operates over the same degrees of freedom, and the computational complexity is unchanged over the run-time. In contrast, the adaptive ROM uses three types of sub-iteration with different memory access patterns. These are identified as the sampled ROM iterations, the sampled basis update iterations, and the sampling and basis adaptation

Algorithm 3 FOM Solver Structure

```
Initialize Mesh X ∈ ℝ^N and state q ∈ ℝ^{N_states} collocated at each volume element
for t ← t_0 to t_M do                                   ▷ Physical time-step loop
    for i ← i_0 to i_M do                               ▷ Sub-iteration loop
        for c ← c_0 to c_N do                           ▷ Loop over cell volumes
            R(q) = f(q, q^{t-1}, q^{t-2})               ▷ Calculate cell residuals
        end for
        Δq ← L(R(q))                                    ▷ Linear solve change in state from residual
        q = q + Δq                                      ▷ Update state for next sub-iteration
    end for
    q_{n+2} = q_{n+1}                                   ▷ Update previous time-steps
    q_{n+1} = q_n
end for
```

iterations.

1. **Sampled ROM iterations:** During these iterations, the hyper-reduced ROM is iterated, and the state at the sampled points ($\mathbf{S}_{n-1}^T \hat{\mathbf{q}}^n$) is updated. These are the most computationally efficient sub-iterations and require no evaluation of the FOM operator. During the initial window of reconstructive ROM, these are the only iterations. After the initial window, these iterations typically represent half of the total sub-iterations per full ROM iteration.

2. **Sampled ROM basis adaptation:** During these sub-iterations, the full state at the sampled points $\mathbf{S}_{n-1}^T \hat{\mathbf{q}}^n$ is updated. After the state is estimated, the basis at these sampling points is updated using the one-step update process. $\mathbf{S}_{n-1}^T \mathbf{V}_p^n = \mathbf{S}_{n-1}^T \mathbf{V}_p^{n-1} + \mathbf{S}_{n-1}^T \delta$ In short, the state and basis at the sampling points are updated during these hybrid sampled iterations. However, these hybrid iterations are still performed on the sampled mesh maintaining the computational savings.

3. **Sampling and basis adaptation iterations:** The final sub-iteration type is one where the sampling and basis update for the full discretization is performed. This will replace the Sampled ROM basis adaption sub-iterations on sampling update iterations. These iterations occur at a frequency defined by the hyperparameter z_s. The ROM state is approximated at all the points. The basis is then updated using the one-step adaptation, and the sampling points are reevaluated based on

their interpolation error. These iterations are the most expensive (on par with the full-order evaluation); ideally, we aim to have as large of a z_s as possible.

The overall interleaving logic of these iterations types is diagrammed in algorithm 4.

Algorithm 4 Adaptive Sampling Solver Structure

```
Initialize Mesh X ∈ ℝ^N and state q ∈ ℝ^{N_states} collocated at each volume element
Calculate initial modal coefficients from initial basis V and sampling points S
for t ← t_0 to t_M do                                   ▷ Physical time-step loop
    for i ← i_0 to i_M do                               ▷ Sub-iteration loop
        for c ← c_0 to c_N do                           ▷ Loop over cell volumes
            if c is sampled then
                R(q) = f(q, q^{t-1}, q^{t-2})           ▷ Calculate cell residuals if sampled
            end if
        end for
        Δq ← L(R(q))                                    ▷ Linear solve change in sampled state from residual
        q = q + Δq                                      ▷ Update sampled state for next sub-iteration
    end for
    q_{n+2} = q_{n+1}                                   ▷ Update previously sampled time-steps
    q_{n+1} = q_n
end for
if mod(t,z_s) == 0 then                                 ▷ Sampling and basis adaptation iterations
    Approximate state at all points
    Update the basis at all points
    Calculate interpolation error at each point to select new sampling points
else
    Approximate state at sampled points
    Update the basis at sampled points
end if
```

5.4 Adaptive Sampling Method Results

The results and capabilities of the adaptive sampling method will now be exhibited. In this chapter the focus is primarily placed on the predictive capability of the method. Practically speaking the adaptive sampling method introduces a number of hyperparameters that are compounded by those that are only relevant to the computer science aspect of the work. Any concerns related to those parameters and the corresponding implementation and scalability of the method will be held until the following chapter. The following results are executed on 2 MPI processes (two parallel memory-independent threads).

5.4.1 Overall Adaptive ROM workflow and pre-processing

The work flow of the adaptive sampling ROM is functionally identical to that of the unsampled adaptive ROM presented in chapter 4. Like the unsampled adaptive ROM of the GTMC, pre-processing requirements are significantly diminished due to the adaptive nature of the method. As a result, the full-order model only needs to be run for a short period to train the ROM. With the introduction of sampling compared to previous results, an initial sampling set is required and computed offline using PLATFORM. Within static sampling studies the choice of sampling point has a large impact on the accuracy of the ROM. For the adaptive sampling method since the points are updated as the ROM advances the initial choice is less critical.

The resulting ROM preparation is as follows:

- Offline
  1. Save a small set of FOM state data $\begin{bmatrix} \mathbf{q}_1 & \dots & \mathbf{q}_M \end{bmatrix}$.
  2. Calculate the centering profile $\overline{\mathbf{q}}$ (average or initial condition) and the primitive ($\mathbf{H}$) and conservative($\mathbf{P}$) normalization constants
  3. Calculate the trial basis $\mathbf{V}$ using the centered normalized field data using POD
  4. Calculate the initial sampling points
  5. Compute $[\mathbf{S}^T\mathbf{U}]^+$ using the trial basis as the residual basis. ($\mathbf{U} = \mathbf{V}$)
- Online
  1. Launch ROM using pre-computed quantities from the training region.

Similar to the GTMC results depending on the problem size and training region, these pre-processing steps can be expensive regarding memory and are overcome using PLATFORM.

Metrics and Hyper-Parameters

Before moving to the adaptive sampling exhibition, the hyper-parameters that can be adjusted for each adaptive ROM case are considered. These parameters are part of the online ROM execution and greatly affect the resulting ROM performance.

- k: The number of orthogonal vectors contained in the trial basis $\mathbf{V}$. The truncation of extra modes for static ROMs sets the predictive capability's absolute upper limit. However, with the adaptive ROMs significantly reduced training time, a significantly higher energy content can be captured with a lower mode count than static methods.

- s: The percentage of total points used for the hyper-reduction of the full dimension M. As a baseline, this is usually chosen anywhere from $1 - 0.01\%$. A higher sampling rate will generally correspond to a more accurate ROM at the expense of evaluation time.

- z_s: The number of iterations between each sampling update step. z_s governs how often the sampled state is used to estimate the full dimensional state, and more importantly, from an efficiency perspective, the full dimensional FOM operator is evaluated.

For completeness, it should be noted that two additional ROM hyperparameters exist, the frequency of basis updates, and the number of DEIM modes. The basis is updated at every iteration for all cases presented in this dissertation. The distinction, compared with the unsampled adaptive basis method shown in the previous chapter, is that for the hyper-reduced ROM, this update step has the same order of computational complexity as the sampled ROM. This makes it anywhere from 10-1000 times faster depending on the sampling rate and problem. Regarding the dimension of sampling modes $(\mathbf{S}^T\mathbf{U})^+ \in \mathbb{R}^{d\times s}$, for all cases presented in this thesis, the number of sampling modes used is the same as the number of trial basis vectors k.

To analyze the resulting performance of the ROMs two metrics roughly corresponding to accuracy and efficiency are introduced. The first of these is the accuracy of the ROM

when compared to the FOM within the predictive region. This metric is similar to the projection error used to analyze the quality of the projection in chapter 4 with the distinction that it is comparing the L2 error of the ROM directly instead of the projection of the FOM.

$$\% \text{ Error} = \left[\frac{1}{MT} \sum_{i}^{i=M} \sum_{t}^{t=T} \frac{||\tilde{\mathbf{q}}_i^t - \mathbf{q}_i^t||_2}{||\mathbf{q}_i^t||_2} \right] \times 100\% \tag{5.17}$$

T is the number of snapshots compared over the testing region, $\tilde{\mathbf{q}}$ is the full state approximation from the reduced order model ($\tilde{\mathbf{q}} = \overline{\mathbf{q}} + \mathbf{H}^{-1}\mathbf{V}\hat{\mathbf{q}}$). This metric aggregates the error between the ROM and FOM over the entire testing window. This quantifies the performance of the ROM in terms of accuracy relative to the equivalent full-order model. The convective type problems that are of interest to use have a variety of local features. As these features propagate relatively small changes in convection speed between the ROM and FOM can translate into extremely large errors. This leads to convective errors being overemphasized relative to source terms. However, the L2 method is the most qualitative method to evaluate the accuracy of the ROM.

To quantify the speed-up achieved by these methods, we compare directly with the equivalent FOM.

$$\lambda = \frac{t_{\text{FOM}}}{t_{\text{ROM}}} \tag{5.18}$$

where t_{FOM} is computational wall time of the full-order model and t_{ROM} is the equivelent wall time of the reduced-order model. Wall time is the time, "clock on the wall," of execution measured from the start to termination of the "online" portion of the ROM. For the rest of this chapter, these two metrics will express the relative success of ROMs. Ideally, the aim is to have as large of a speed-up λ for a given error.

5.4.2 1D Propagating Laminar Flame

The first test case exhibited is that of a 1D flame. The setup has been used for ROM performance and validation cases [55, 53]. The 1D flame is a relatively small problem when compared to the scale and complexity of the GTMC. However, 1D propagation of

the high-gradient flame has been challenging for ROMs. The primary convective feature has provided significant challenges for a static basis method. Additionally, the non-linear term hosts the same complexity and numerical stiffness level as those seen in more complex problems.

The computational domain of this problem consists of a straight duct 10 mm long and discretized into 1000 elements. The domain is considered one-dimensional from a physics perspective but is discretized as a set of two-dimensional quad (4-sided) elements arranged in one dimension. The computational domain is visualized in Fig. 5.2.

The boundary conditions for the setup are slip walls for the upper and lower boundaries. The inlet and outlet are set via a specified characteristic boundary condition. A perturbation of the acoustic characteristic is additionally imposed at the outlet to introduce an oscillation as a sinusoid centered around the outlet pressure.

This perturbation is set as

$$p_{u-c} = p_{u-c,ref}[1 + A\sin(2\pi f t)] \tag{5.19}$$

For the baseline case, an amplitude $A = 0.1$ and a frequency $f = 50000$ Hz. For the reaction effects, the chemical kinetics are simplified. Instead of real species, the reactant and product gases are treated as calorically perfect gases with identical thermodynamic and transport properties. These values are a molecular weight of 21.32 (g/mol), $c_p = 1.538$ $\frac{kJ}{kgK}$,and $\mu_{ref} = 7.35 \times 10^{-4}$ $\frac{kg}{ms}$. The reacting effects are achieved via the difference in reference enthalpy between the product and reactant gases with a value of $-10,800$ and $-7,432$ $\frac{kJ}{K}$, respectively.

The solution is computed using the GEMS solver with second-order accurate dual time-stepping formulation with a constant $\Delta t = 0.01$ μs with ten inner sub-iterations.

Training

The FOM is initialized with a region of cold reactants and hot products with the interface between them at $0.25L$ of the duct length at $t = 0$ μs. The FOM is advanced from the starting IC to $t = 25$ μs. At this point, we begin the ROM training portion described

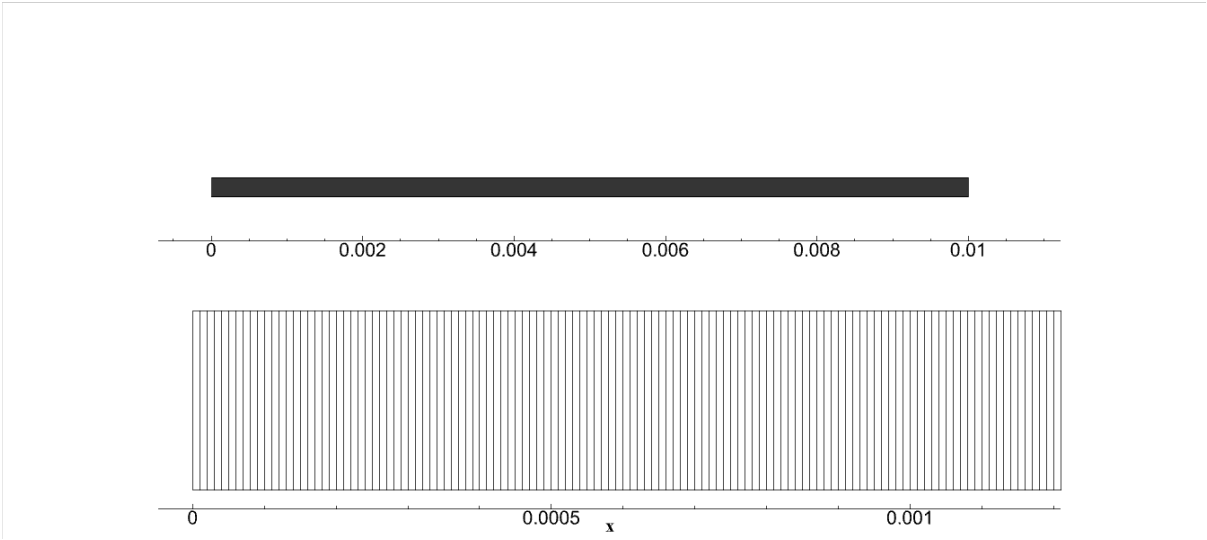


Figure 5.2: Computational domain for the 1D propagating flame: full domain (Top), zoomed in view (Bottom).

above. The FOM is advanced 100 iterations leading to a training range from $t = 25 - 25.1$ μs. This training region corresponds to approximately 1/10 of the induced oscillation frequency. The training set consisted of 10 snapshots, each 10 iterations apart.

From here, the normalization constants are computed from the recorded FOM states. The initial condition of the training region is used to center the data, and the primitive and normalization constants are calculated using the L2 norm method. As done previously, the velocity was normalized based on the velocity magnitude, and the species mass fraction primitive states were normalized by 1.

The trial basis $\mathbf{V}$ consisted of two modes $k = 2$ and the initial point selection was done using the GappyPOD+Eigenvalue method described in section 2.3.6. The set of ROMs was initialized to run from the begining of the training period $t = 25$ μs to $t = 60$ μs. The ROMs were run with sampling rates s set to 1, 5, 10, 20, and 100 % of the total cell count. These correlate to sampled cell counts of 10, 50, 100, 200, and 1000 of the total 1000 cell mesh. As a point of reference, an unsampled ROM would be the equivalent of the 100% sampling rate. The number of sampled cells can vary slightly based on the required additional flux and gradient cells added after the chosen core cells. Finally, the adaptation frequency z_s was tested for 2, 5, 10, and 20. Recall that a sampling adaption rate (z_s) of one would be the equivalent of solving the FOM, as the adaptation

requires evaluating the FOM operator at all mesh points. The computational efficiency is gained via the reduction of cell evaluations. Thus, if we were unconstrained by accuracy, we would want as small of a sampling rate (s) and as large of a sampling adaptation frequency(z_s) as possible.

Performance

To exhibit the dynamics of the flame, we visualize the evolution of the full-order model within the testing period in Fig. 5.3. We can track the convection of the flame front as it moves through the duct toward the outflow boundary condition. Additionally, we can observe pressure oscillation within the duct and the interaction with the flame front as kinks in the pressure profile.

To compare, we visualize the same time instances for the ROM (figure 5.3). The ROM visualized is for the $z_s = 5$ case with 1% sampled points.

This qualitative comparison shows a good agreement with the full-order model. To more directly observe the accuracy, the various z_s value profiles are plotted at $t = 30$ μs and $t = 60$ μs in figure 5.5and 5.6 together with the FOM profile. $t = 30s$ is $5\mu s$ from the start of the testing period and $t = 60$ μs is at the end of the testing period. Here the deviations of the ROM are more observable. As expected, as z_s increases, we see a greater deviation from the full-order model. For $z_s = 2$, the ROM strongly agrees with the FOM over the entire domain at both time instances. For $z_s = 5$ the ROM shows good agreement at $t = 30\mu s$ but shows a phase shift when compared with the FOM by the end of the testing period. This is remarkably similar to the behavior of the static ROMs on the GTMC. However, the flame front convection is still captured quite well. For $z_s = 10$, there is a significant deviation in the region directly after the flame front. This error takes the form of spurious oscillations and an offset of around 20 kPa. As the flame front advances, these oscillations remain contained to the region directly after the flame front. The $z_s = 10$ case is also the only one showing significant flame front variation. At the start of the testing region, the flame front matches for all z_s values. At the end of the training, other than $z_s = 10$, all ROM flame fronts show good agreement. For $z_s = 10$

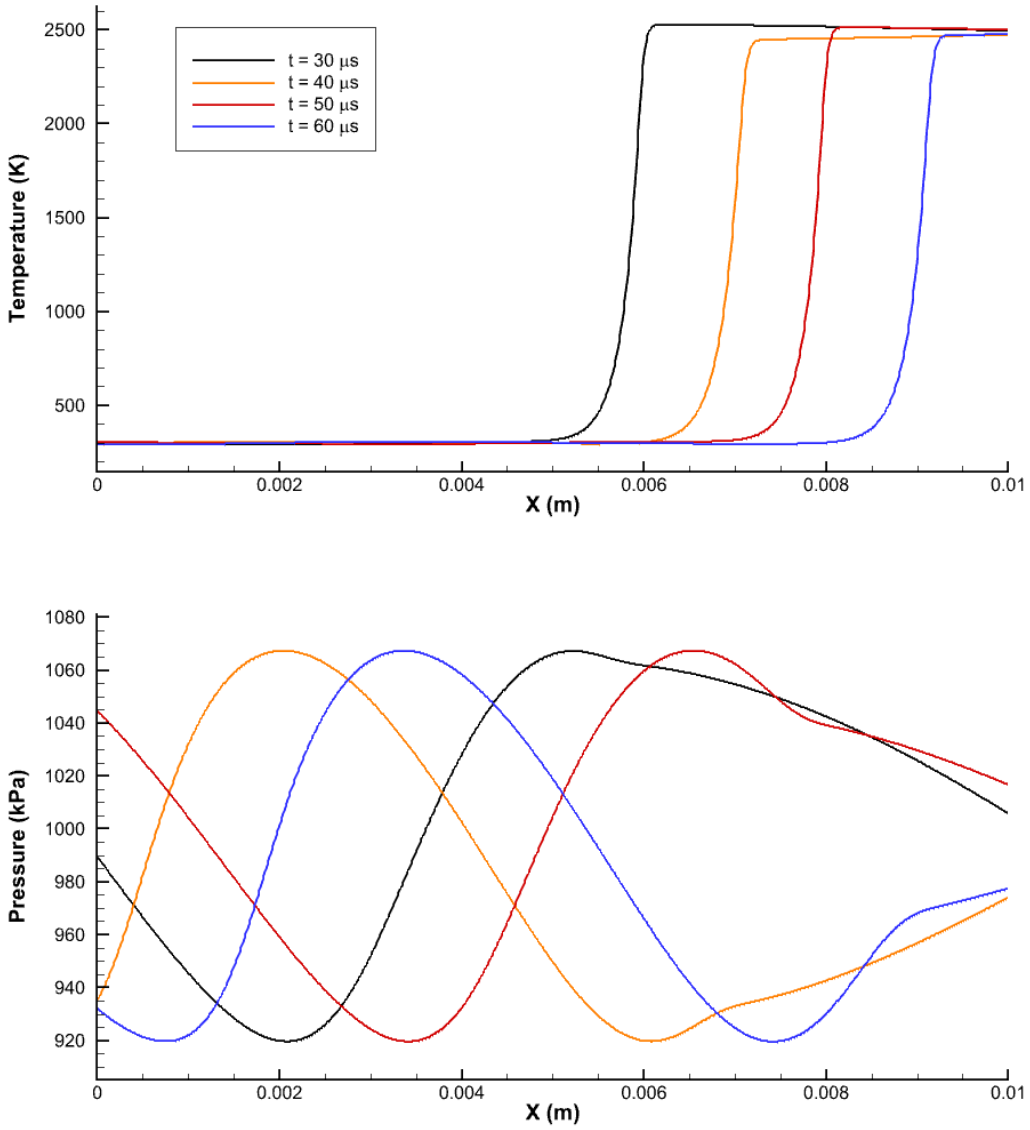


Figure 5.3: Representative profiles for the 1D laminar flame propagation full-order model.

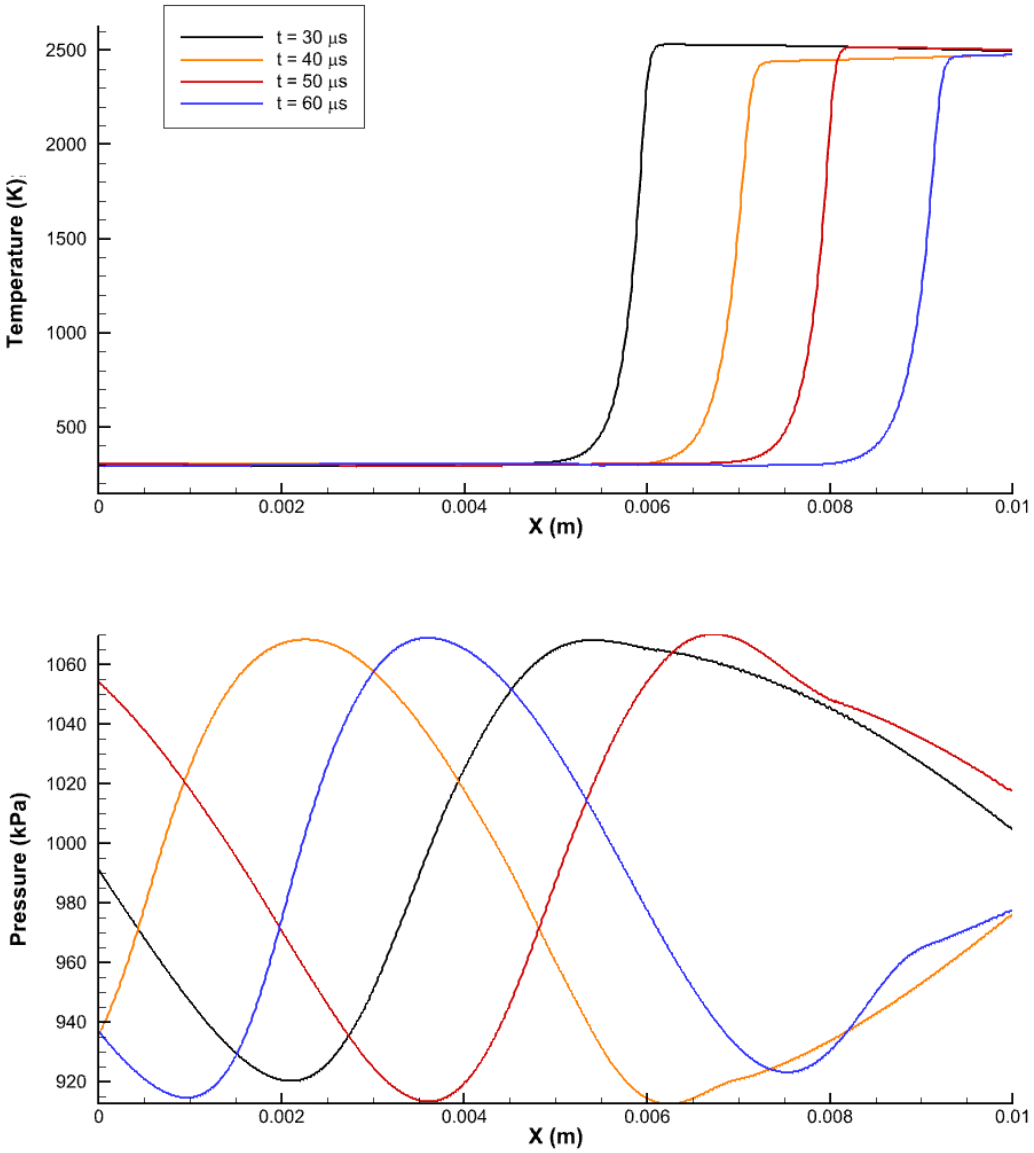


Figure 5.4: Representative profiles for the 1D laminar flame propagation reduced-order model (Right). The reduced order model was trained from $t = 25 - 25.1$ μs.

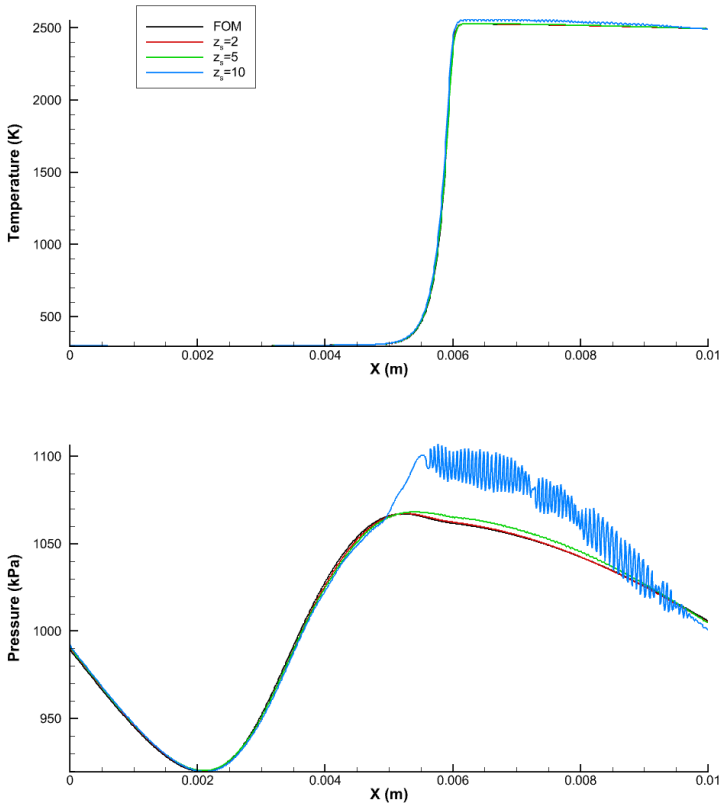


Figure 5.5: Comparison of different sampling update frequencies at $t = 30$ μs.

the flame front has begun to lag from the FOM solution.

So far, we have discussed the deviations of the ROM from a quantitative perspective. Returning to the accuracy and efficiency metrics defined earlier, we now look at the relative performances. The error and speed-up are plotted vs. sampling rate for various z_s values in figure 5.7. We observe that most of the computational efficiency is tied to the sampling rate. This speed-up is multiplied at higher z_s values as the number of full-mesh FOM iterations is reduced. However, we observe in Fig. 5.5 and 5.6 that at higher z_s values, the accuracy of the ROM is compromised. Thus the general rule of thumb is to improve the computational efficiency as much as possible via increasing z_s and decreasing the sampling rate but not so much as to compromise the predictive accuracy. Like the unsampled GTMC previously discussed at a sampling rate of 100%, unsampled, the sampled ROM is slower than the corresponding full-order model.

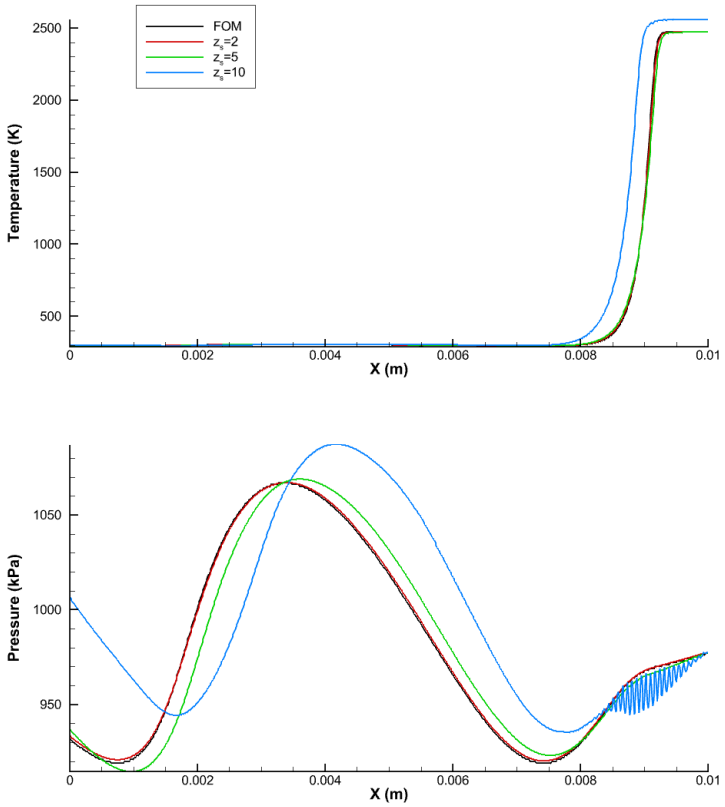


Figure 5.6: Comparison of different sampling update frequencies at $t = 60$ μs.

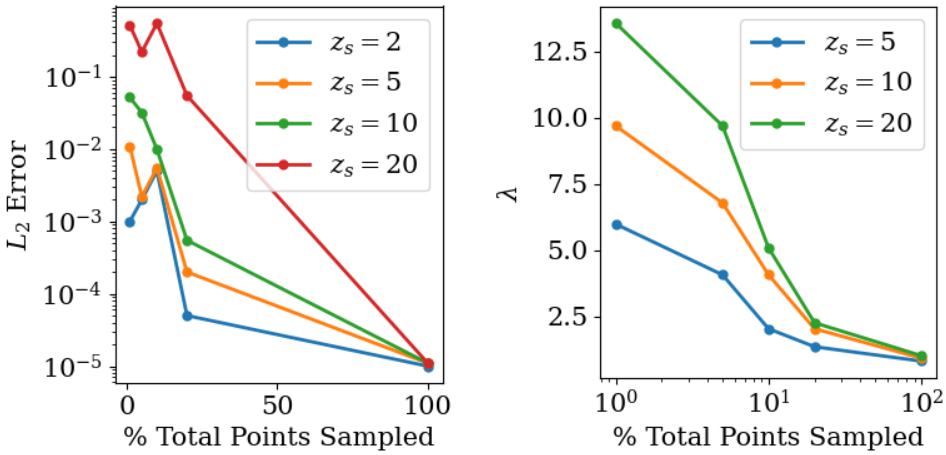


Figure 5.7: L_2 Error compared with full-order model (left) and achieved efficiency (right) for various sampling update frequencies and sampling percentage.

Comparison to Static Basis ROM

To further emphasize the capability of the adaptive ROM compared with the static basis, we plot the state profiles of pressure and temperature in figure 5.8. Here, it is apparent the static ROM cannot properly capture the dynamics, and in particular, the expected convection of the temperature gradient is effectively stationary in the static ROM. On the other hand, the adaptive ROM can predict the system dynamics successfully. The adaptive basis method is critical for capturing dynamics beyond the training region for convective problems. The static ROM in particular shows spurious oscillations both near the flame front and near the outflow boundary condition.

Parametric Capability

To examine the predictive capability of the adaptive ROM, we initialize a ROM with a modification to the outlet forcing frequency. Using the same training leveraged in the previous 50kHz results, we run the ROM with a 200kHz forcing. The ROM can predict this transient response well and captures the change in pressure wave in the domain. As a result, this shows the significant capability of these ROMs for the many query applications prevalent in the design environment. Considering the performance increase, we observed in figure 5.7 5-10 times as many parametric ROM cases can be run in the same length of time as a single full-order model.

5.4.3 2D Rocket Injector

The second baseline case we will use to evaluate the performance of the adaptive sampling method is the 2D rocket injector. Similar to the 1D laminar flame, this case has been used extensively in developing and quantifying ROMs [55, 88].

Problem Description

This geometry is representative of a simplified laboratory-scale rocket combustor. The computational geometry is shown in figure 5.10. The system comprises a central oxidizer

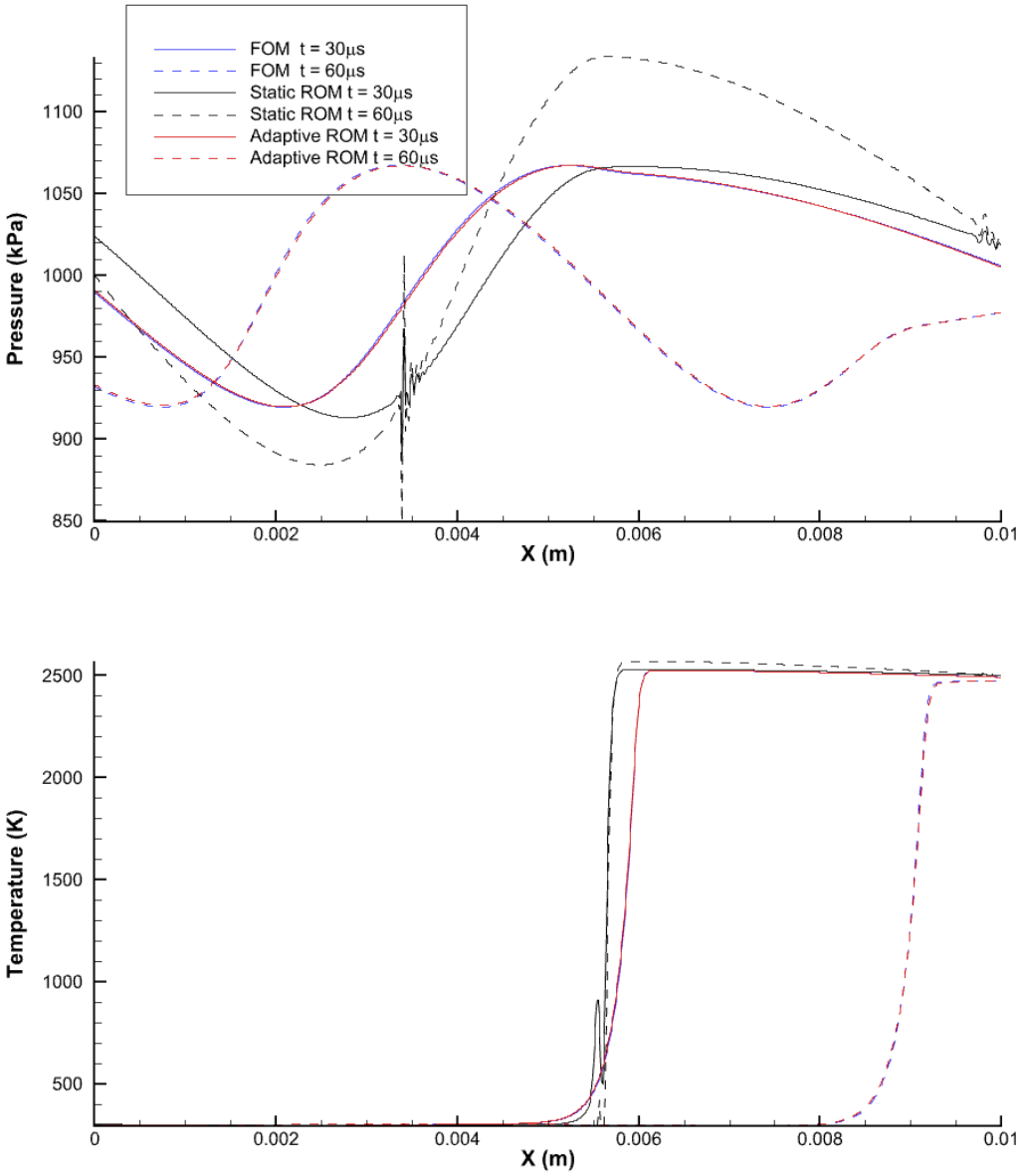


Figure 5.8: Comparison of the adaptive ROM vs. Static and $t = 30$ μs and $t = 60$ μs, The reduced order model was trained from $t = 25 - 25.1$ μs.

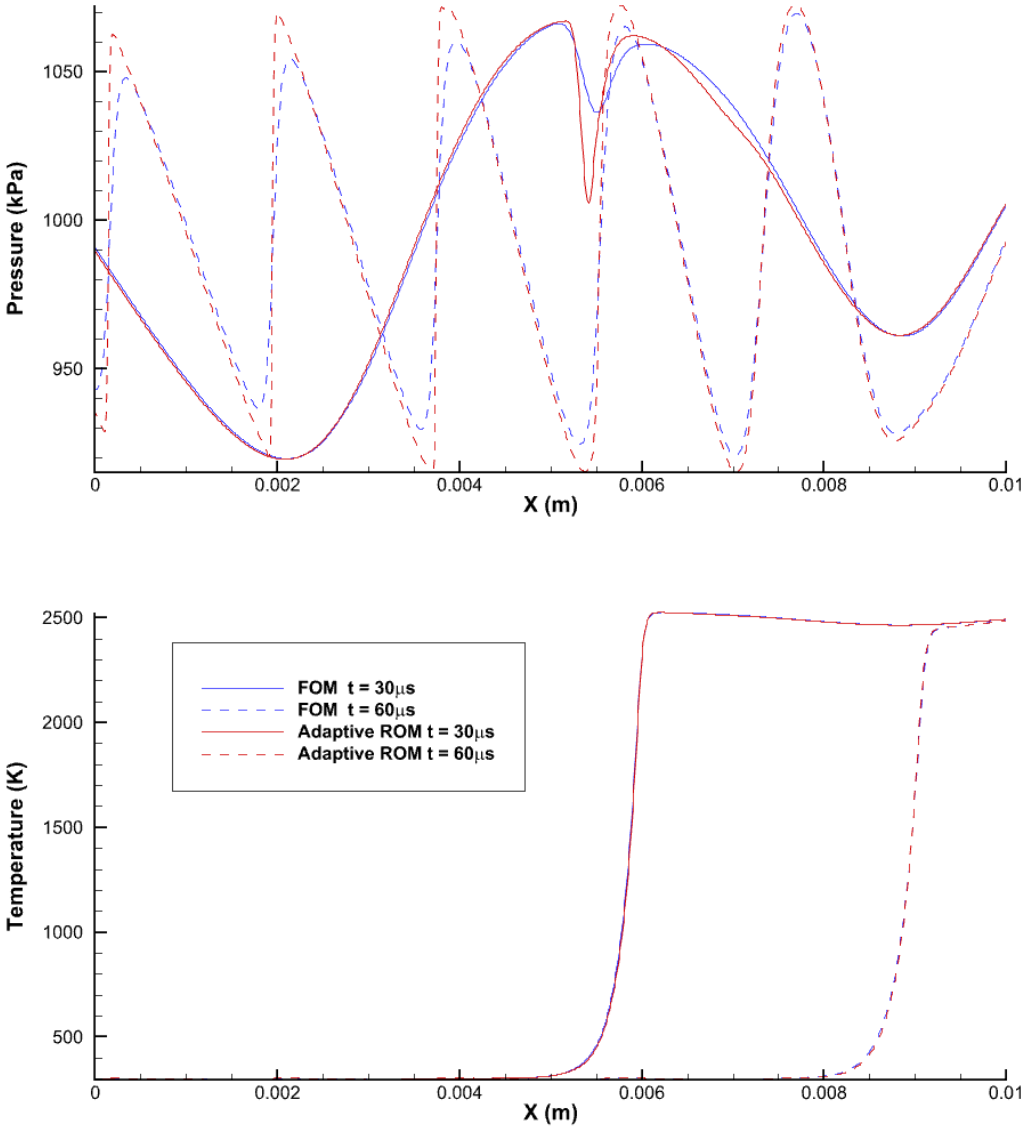


Figure 5.9: Comparison of the adaptive ROM and FOM for a parametric case where the forcing frequency is increased from 50kHz to 200kHz. Training Window: $t = 25 - 25.1\mu s$.

	$\dot{m}$ kg/s	T (K)	Y_i
Oxidizer	5.0	700	$Y_{O_2} = 0.42$, $Y_{H_2O} = 0.58$
Fuel	0.37	300	$Y_{CH_4} = 1.0$

Table 5.1: 2D injector baseline operating conditions.

inlet with a coaxial fuel dump plane which terminates into a larger injector chamber. The inflow conditions and mass flow are presented in table 5.1.

The overall conditions are characteristic of a laboratory combustor and correspond to an adiabatic flame temperature of 2700K. The outflow is perturbed with an acoustic oscillation of identical character to that used in the 1D laminar flame with an amplitude of $0.1\bar{p}$ and frequency of 5000 Hz.

For the reacting physics, the finite rate chemistry model is used with a simplified 1-step methane oxygen reaction of Westbrook and Dryer [89] with a pre-exponentiated factor reduced by order of magnitude. These correspond to $A = 6.7 \times 10^{11} \frac{\text{gmol}}{\text{cm}^3}^{1-a-b}$ with reaction exponents of $a = 0.2$ and $b = 1.3$.

$$\text{CH}_4 + 2\text{O}_2 \rightarrow \text{CO}_2 + 2\text{H}_2\text{O} \tag{5.20}$$

Each of the gases is considered a thermally perfect gas with their thermodynamic and transport properties calculated via the NASA polynomial [60] and the Wilk and Mathur transport coefficients [90, 91] as discussed in chapter 2.

The overall physical system has two primary dynamics. The first of these is reaction physics characterized by the shear layer hosting a range of spatiotemporal scales due to the reaction physics, eddy roll-up, and larger recirculation region near the backward-facing step. In addition, the pressure signal forms a waveform corresponding to the induced pressure oscillation from the outlet. In the context of a rocket injector, this pressure oscillation is similar to the expected outflow condition from the larger rocket combustion chamber. As a result of these interacting dynamics, the range of scales, chemical reactions, and especially the resulting chaotic dynamics, this system makes an excellent case study for predictive ROM development.

The full-order model was restarted from a previous FOM computation by interpolating

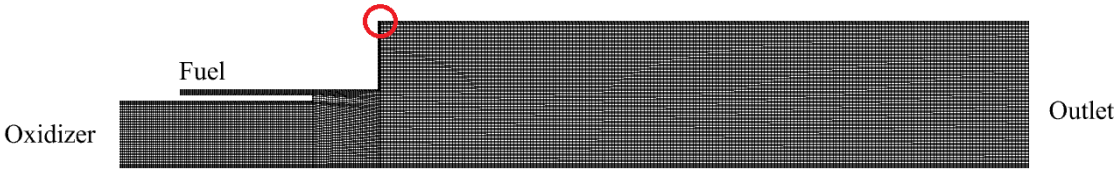


Figure 5.10: Computational mesh for the 2D rocket injector with point monitor probe identified with red circle.

the field onto a new partitioned layout of 2 processors. The computational domain consists of 38,523 quad elements and, for a FOM solve, would nominally be partitioned onto approximately 40 processors with $O(1000)$ cells per process. For the baseline study, the mesh is split among 2 processors with $O(20,000)$ cells per processor. For the 2D baseline timings and efficiencies, the cases are run on a single ERDC Onyx node. Each Onyx node comprises two 22-core Intel E5-2699v4 Broadwell processors for 44 available cores.

The solution is advanced using the dual time-stepping formulation with a $\Delta t = 1 \times 10^{-7} s$ for $0.0029s$ allowing any interpolation errors from the repartitioning to dissipate. The training region is set from $t = 0.0029 - 0.00291s$, with a testing region from $t = 0.0029 - 0.0049$. A sequence of FOM snapshots of temperature and fuel mass fraction for the testing region is shown in figure 5.11.

A similar training set is established using the same process as for the 1D benchmark case. The FOM is run for 100 iterations saving every ten snapshots resulting in 10 snapshots at $1 \times 10^{-6} s$ intervals. These snapshots were then centered by the initial condition and normalized. These ten snapshots produce 9 non-zero basis modes (due to the initial condition centering).

For the baseline ROM we use 5 of these basis modes to construct the trial basis $\mathbf{V} \in \mathbb{R}^{M \times k}$. The initial sampling points are selected with a 1% sampling rate. The FOM and ROM are run with 10 sub-iterations per physical timestep, and half of the ROM sub-iterations are used for basis adaptation.

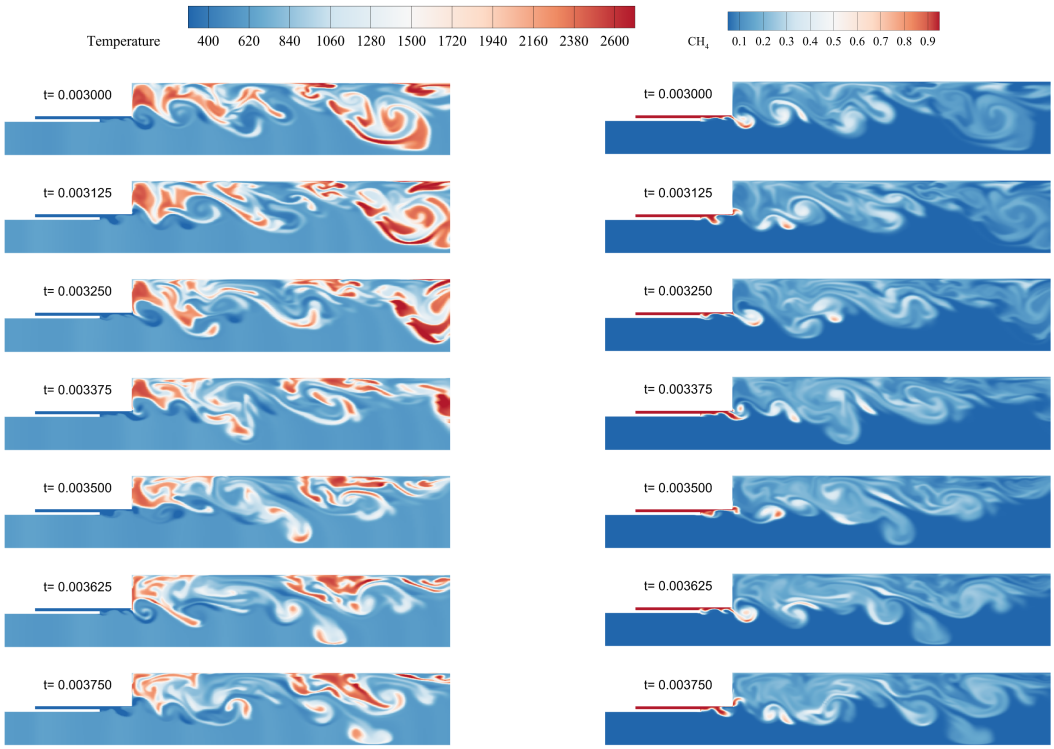


Figure 5.11: 2D injector FOM representative dynamics with temperature (left) and methane mass fraction (right) at various time instances.

Performance

Like the 1D case, the system evolution is compared qualitatively for a ROM with a sampling adaptation rate $z_s = 10$. In figure 5.12 we show the evolution of the temperature field at the same time instances in the testing region as the FOM shown in figure 5.11 with the field error for comparison. We can observe identical dynamic features between the FOM and the ROM, most notably the shear layer roll-up and convection downstream, with local regions of high-temperature matching qualitatively. Examining the error fields on initial impression, there are regions of significant error. However, most of this results from convective features being offset spatially. This spatial offset, combined with the sharp gradient of the reacting flow, leads to large spatial errors in these regions.

Figure 5.13 shows a probe monitor of the pressure signal for a more quantitative comparison. The location of the probe is identified in Fig. 5.10. Here we see that the ROM shows excellent agreement for both adaptation frequencies. In addition to the expected oscillations from the outlet forcing, the FOM still captures non-periodic effects

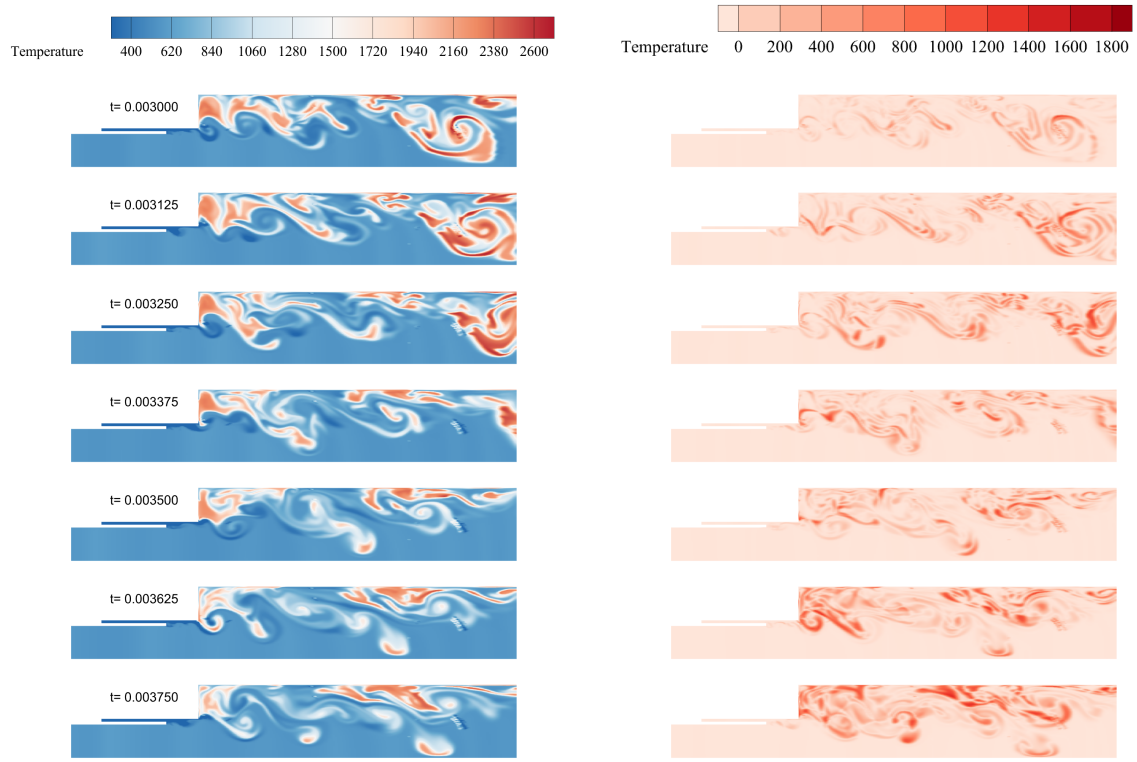


Figure 5.12: 2D injector ROM temperature field (left) and error (right) for $z_s = 10$. Training region ($t = 0.0029 - 0.00291$s).

as observed by the mean increase and decrease of the oscillations. These effects are due to the combustion dynamics and are captured well. Finally, it must be noted that the training region is 2% of the total test region. The information contained in the training region is less and contains a tiny portion of even a single high-frequency cycle. This feature makes the adaptive ROM attractive as the training requirements are very light.

We now quantify the total performance of the adaptive ROM for the hyperparameters of the sampling rate and the sampling adaptation frequency. Figure 5.14 shows the L_2 error (defined by the total field error over the training region) and the efficiency λ for the various sampling rates at various adaptation frequencies. Here we observe that across all z_s values, the primary improvement in speed up is via the hyper reduction sampling. The efficiency and field error increase as the number of sampling points is reduced. For both $z_s = 5, 10$, the field error remains less than 1% for the entire range of sampling rates. We observe that at the higher sampling rate of 20, the error is significantly higher at almost 52%. This behavior is expected as if the sampling points are not updated frequently

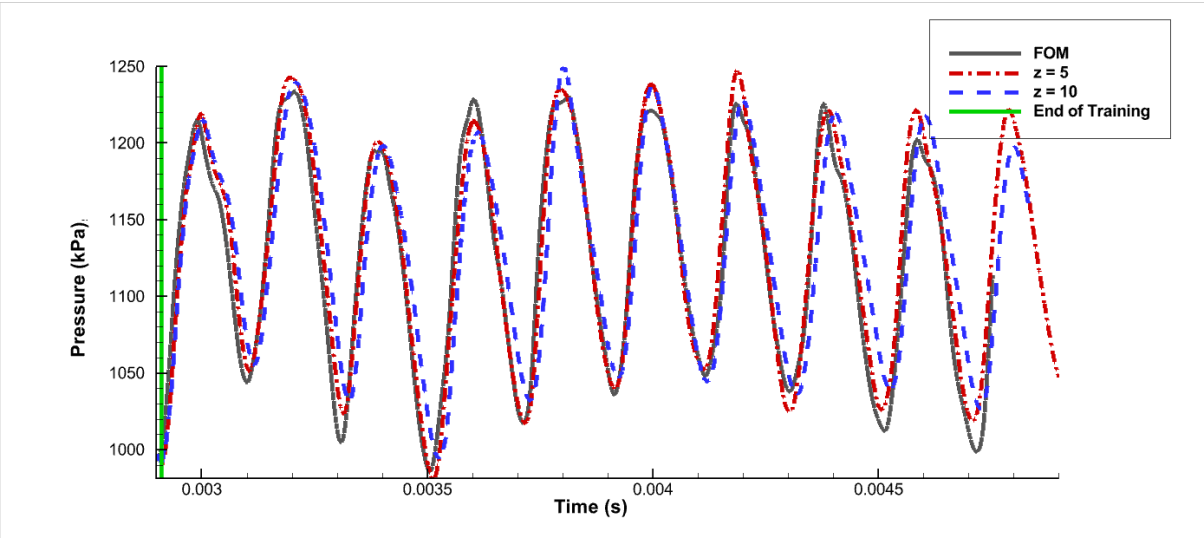


Figure 5.13: Pressure signal comparison of FOM and ROM for various sampling adaptation rates with a sampling rate of 1% in the predictive region. Training region ($t = 0.0029-0.00291$s).

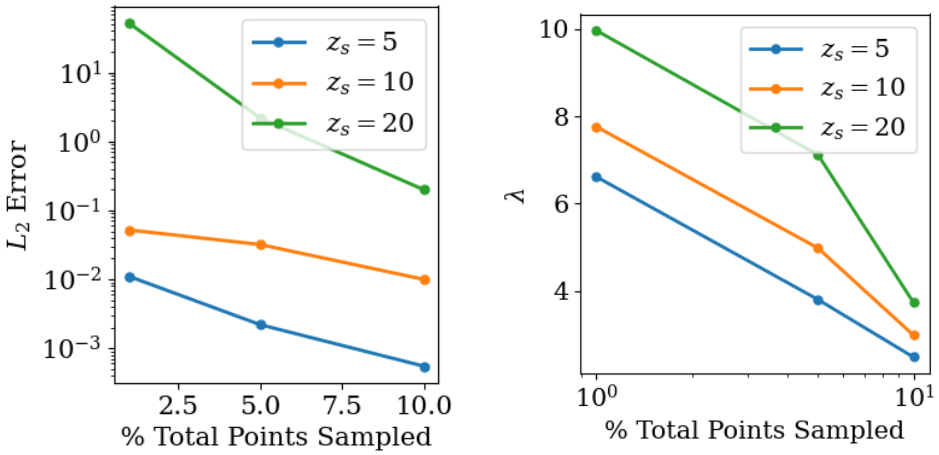


Figure 5.14: L_2 error compared with the full-order model (left) and achieved efficiency (right) for various sampling update frequencies and sampling percentages.

enough, the build-up of error becomes too large for the full FOM evaluation to rectify. As a result, for this type of adaptive ROM the key hyperparameter tuning is based on choosing the lowest sampling rate and largest sampling update rate that allows the ROM to remain predictive to achieve the greatest computational efficiency.

Comparison to Static Basis ROM

To show the behavior of the adaptive ROM (fig. 5.12) compared with a static basis, we first execute a static ROM with the same training period as the adaptive ROM. In figure 5.15, we compare a static ROM using the same training window as the baseline

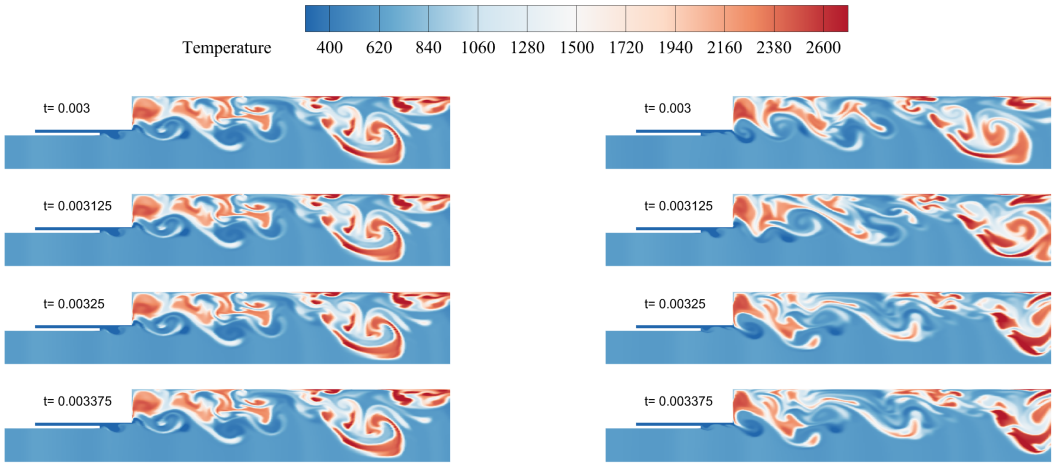


Figure 5.15: 2D injector static basis ROM temperature with two modes with identical training ($t = 0.0029 - 0.00291s$) as adaptive ROM (left) and 50 modes with training ($t = 0.0029 - 0.00325s$) (right).

adaptive ROM and 50 modes static basis trained from $t = 0.0029 - 0.00325$. Here we observe the same behavior as in the GTMC. The static basis ROM can capture dynamics within the training region but struggles significantly beyond this range. For the adaptive type training, this causes the ROM to freeze effectively in place since it cannot replicate the dynamics using 2 modes with such a small training window. For the longer training window, similar to the GTMC, the static ROM can accurately capture the system's evolution until the end of the training reconstruction, where it freezes with some smearing of fine-scale features.

For both the 1D and 2D cases, the adaptive ROM significantly improves predictive accuracy. This is unsurprising as the basis adaptation method has already proven its capabilities, as shown on the GTMC. The final piece, however is the application of sampling to achieve computational speed-up relative to the FOM. Even with relatively high sampling rates and sampling adaptation rates, the ROMs are able to stay reasonably accurate and predict dynamics consistent with the FOM.

5.5 Tertiary usage of adaptive ROMs in larger frameworks

Until now, the focus has been placed on the capability of the ROMs. The FOM LES simulations have acted as the ground "truth," and the ROM methods are "successful" when they can replicate the results and are practical if they also provide a meaningful computational speed-up.

However, there are scenarios where FOMs cannot be eschewed. For example, high-fidelity simulations often validate proposed physical models against experiments. In this scenario, the use of projection-based ROMs would be unwise as the modeling error would be indistinguishable from the ROM error (either projection or time integration error). Alternatively, the use of high-fidelity simulation may be mandated by program requirements.

Within the work in this dissertation, high-fidelity simulations (LES) of the complex GTMC system were conducted. After the required offline mesh generation steps were completed, the online computation was started. The simulation was initialized using relatively non-physical initial conditions and allowed to reach a statistically stationary state. This initial transient contains very little useful information and in fact veils any potential problems. In the GTMC case, these transients required equal resources as the time period used to collect statistics. After an initial examination, it was realized that some changes to the boundary conditions and the mesh were required. Thus the entire transient had to be evaluated with the new setup. The transient can be somewhat diminished by using previous calculations as the initial condition; however, depending on the domain size and problem characteristics, these transients are still a waste of valuable computational time. An example evolution of such "numerical" transient of this type is shown in Fig. 5.16 for the 2D rocket injector.

Some more production-minded solvers have methods to alleviate this. The most common is an in-situ mesh refinement. This method initializes the solver on an extremely coarse mesh which diminishes the computational cost of this initial transient. However,

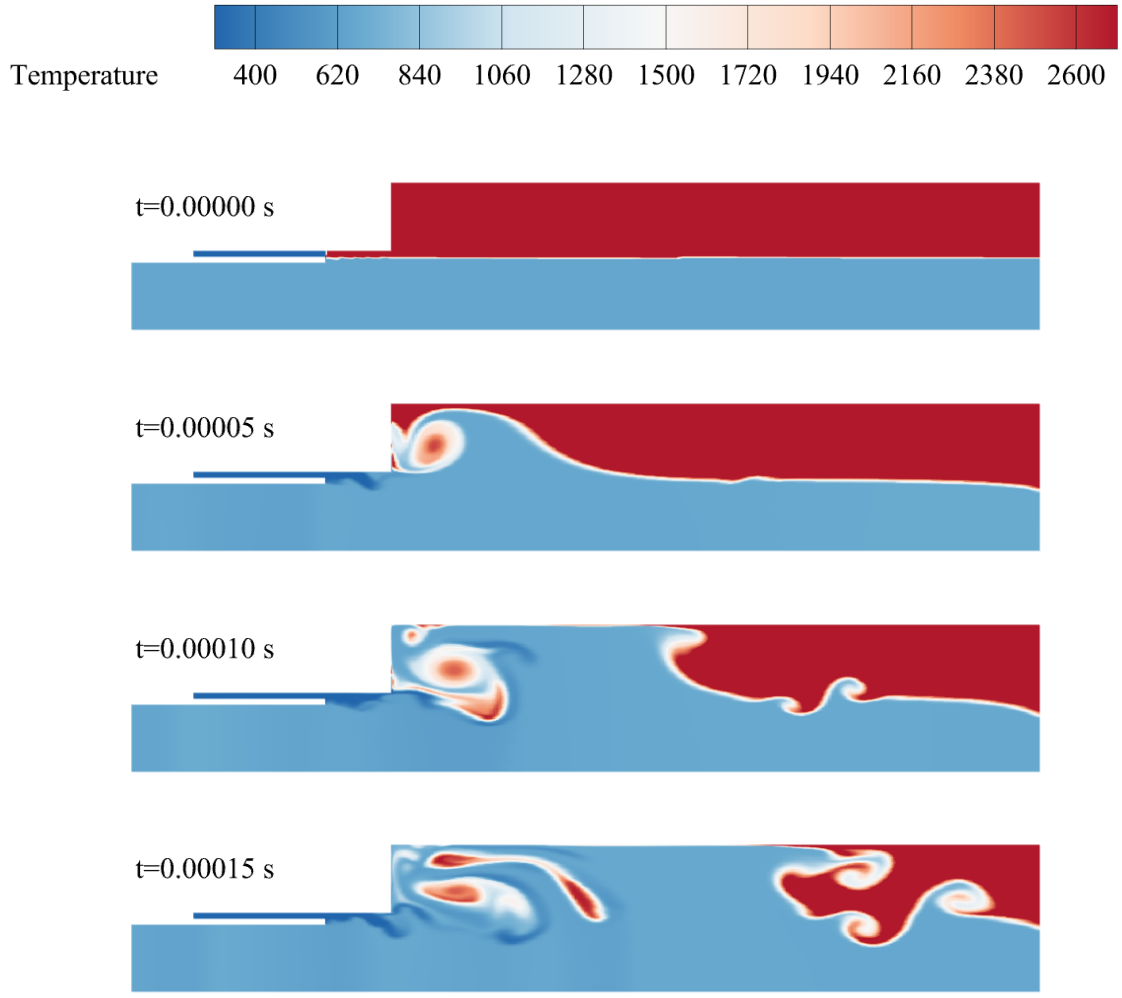


Figure 5.16: Temperature Contours of the 2D rocket injectors numerical transient from initial conditions.

these methods are still fundamentally limited by the numerics of the physical system, unlike projection ROMs. Additionally, it is unlikely that a coarse mesh would have even close to the computational speed-up possible with ROM. The concept of in-situ mesh refinement will be revisited later in the context of the computational challenges associated with adaptive sampling.

5.5.1 Transient Acceleration: 1D Flame

A method using these adaptive ROMs to accelerate these computational transients is easily envisioned. The potential error as a result of the sampling or projection error is even less of a concern in these instances, so an even more computationally efficient ROM can be constructed as long as they remain numerically stable.

To exhibit this capability we use the 1D propagating flame used previously. The initial condition for this problem is a sharp flame front placed at 25% of the domain length with a uniform region of pressure, velocity, temperature, and composition on each end. The pressure oscillation is not present as it is introduced through the outlet boundary condition as the flow develops.

We use the same procedure as the baseline predictive cases and train the initial ROM basis on a set of snapshots produced just after the FOM initialization. The FOM solver uses a ramping CFL number to ease any numerical instability due to the non-physical initial conditions. We wait for this ramping to end and then take 10 snapshots over the 100 timesteps. We then initialize the ROM with a sampling rate $s = 1\%$ and a sampling update $z_s = 20$. Recall that the accuracy of this transient is not critical. The transient response is plotted in Fig. 5.17 for the transient from the FOM and adaptive ROM. Given the aggressive construction of the ROM the transient profile shows deviation from the FOM that would be problematic from a pure prediction standpoint; however, trends in the correct direction result in the same statistically stationary flow. For both for z_s equal to 5 and 10 the end of the transient region would be acceptable initial conditions which were computed at 1/6th and 1/10th of the FOMs cost, respectively. These savings would be even more significant for larger problems. However, more complex problems would

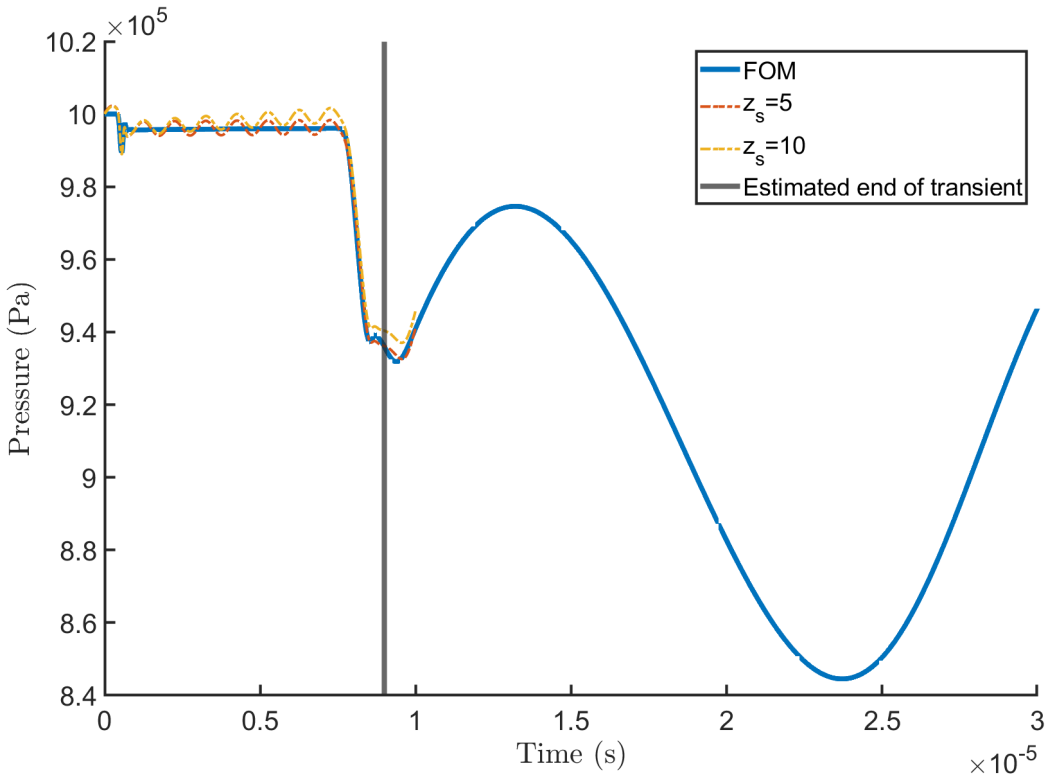


Figure 5.17: Example application of ROM to accelerate the initial transient of the 1D flame problem.

presumably have much tighter robustness windows on the allowable z_s value.

5.6 Summary

Adaptive ROMs are shown to be a powerful predictive tool to significantly reduce the computational cost of modeling complex multi-scale transport problems. At a fraction of the training requirements of classical static bases ROMs these adaptive methods are capable of both predictive and parametric use. In addition to being a valuable stride in the development of projection ROMs for use in predictive and parametric use.

Chapter 6

Adaptive Sampling: Computational Considerations

6.1 Outline

The adaptive basis method has shown a predictive capability far beyond that of static basis methods. At the cost of periodic, full-order evaluations the basis and sampling points are updated to allow greater stability and accuracy together with computational efficiency.

The sampling update step in the adaptive sampling formulation requires evaluating the FOM operator on the full mesh. This means that computationally both the sampled and unsampled mesh must exist in memory. In the baseline implementation, these create an additional hidden parameter in the form of the partition count. If we run on a small number of partitions, we achieve the goal of reducing the computational requirement; that is, a FOM that might require an HPC compute node could be downsized and run as quickly as an adaptive ROM on a personal laptop. However, this will increase the evaluation time of the sampling update step relative to running on the original larger FOM processor count.

6.2 Description of Integration with Solver

In the previous chapter, we have described the algorithms and methods we wish to apply to a reacting flow CFD solver. However, significant computational challenges must be overcome to realize theoretical computational efficiency. This chapter will discuss the

computational challenge of integrating the scalable, adaptive sampling ROM method into an existing CFD solver in conjunction with the strategies applied.

Let us first consider the computational methodology of the solver when solving the FOM. The quantities of interest are at each discretization point for a given spatial discretization of a problem. These quantities are located on the volume "cell" elements for a finite-volume solver. The governing equations are applied to the mesh. This leads to a system of ODEs that describes the evolution of the system. The resulting linear system can then be integrated to predict the evolution of the physical system. Each element (spatial discretization, application of governing equations, time integration) is a significant area of study.

$$\mathbf{f}(x) = \mathbf{f}_{\mathbf{flux}}(\mathbf{q}(x, x \pm \Delta x)) + \mathbf{f}_{\mathbf{source}}(\mathbf{q}(x)) \tag{6.1}$$

The changes in the state of a point are defined by the quantities of interest within the state (usually represented as a source) and its surroundings (fluxes). This area of influence is visualized in Fig. 6.2. As a result, the computational cost of modeling a given physical system directly scales with the number of degrees of freedom (DOFs). These high-dimensionality systems can emerge in two major ways. The first is a significant number of degrees of freedom that can be present at each discrete physical point. Reacting systems are an excellent example, as even relatively elementary reactions depend on many chemical species. Alternatively, the system of interest may require a fine discretization to represent the physical system properly. Turbulence modeling requires fine discretizations to properly model the turbulent energy cascade, especially near physical walls. These high DOF systems are extremely expensive to simulate as the required floating point operations (FLOPs) often scale exponentially with additional degrees of freedom.

Most CFD solvers achieve scalability on high-performance computers (HPC) via domain decomposition. The physical mesh is decomposed into roughly equal partition structures for a given computational domain (figure 6.1). Each partition is advanced individually, with each processor taking on a fraction of the required DOFs. However, to maintain the global solution a communication step between the various processors is

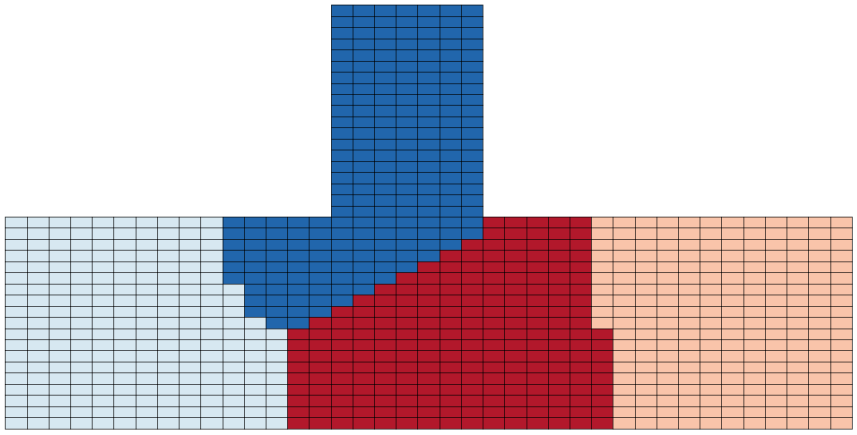


Figure 6.1: Example representation of a four-way partition of a 2D pipe junction.

required to allow information about physically local DOFs to influence each other.

6.3 Description of Existing Adaptive Sampling Method

There are two major loop structures as part of the implicit solve the time step iteration and the sub-iterations of the linear solve using pseudo-time-stepping. The FOM algorithm as outlined in the discussion of the adaptive sampling method (Algorithm 3). Here all N cell elements of mesh X are iterated over to compute the corresponding residuals and change in state q. The scaling is more apparent if we expand the residual calculation as shown in algorithm 5.

Here it becomes more apparent how the number of computations scales with the

Algorithm 5 Residual Calculation

$R(q) = \mathbf{f}(q, q^{t-1}, q^{t-2})$ ▷ Calculate cell residuals (2nd order time)
for $f \leftarrow f_0$ to f_N **do** ▷ Loop over Faces
  flux $= \mathbf{f}(q_L, q_R)$ ▷ Compute flux face from the state of adjoining cells
  flux = flux $+\mathbf{f}(q_{l_{neighbors}}, q_{r_{neighbors}})$ ▷ Higher order flux
  $\text{residual}_L = \text{residual}_L - \text{flux}$
  $\text{residual}_R = \text{residual}_R + \text{flux}$
end for
for $c \leftarrow c_0$ to c_N **do** ▷ Loop over Cells
  residual = residual $+ f(q_c ell)$ ▷ Compute Source Contributions
end for

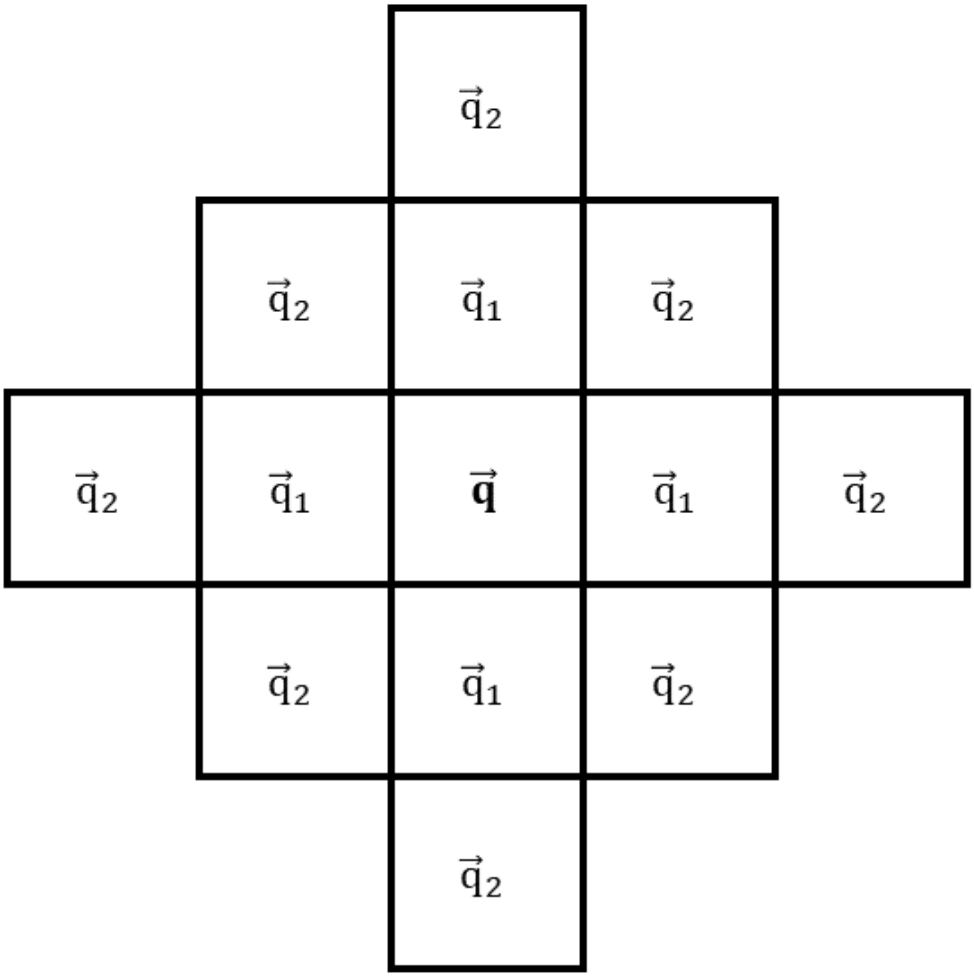


Figure 6.2: Example of the cell information needed to compute the residual of a given cell for a 2nd order spatial scheme.

overall dimension of the mesh X as not only does the solver have to loop over each cell to compute the source term, but it also must iterate over all the faces to compute the flux contributions. In addition, we need to calculate the gradients for higher-order spatial schemes such as the ones we will be using. The ultimate goal of model-reduction and, more specifically, this work, hyper-reduction, is to reduce the number of evaluated cells significantly.

6.3.1 Description of the Partitioning Strategy

For an HPC scale fluid solver, partitioning is critical. Practically speaking, the first step in the workflow is partitioning a computational mesh composed of N elements. In the GEMS solver workflow, this is done using a METIS [92] based serial partitioner. The cell map is sliced into P partitions with approximately N/P number of discretized elements on each partition. Once this partitioned cell map is generated, the communication layout between the adjoining partitions must be constructed. Generally, this communication layout can be done in various ways depending on the solver stencil. A common way this is done is via communication of an "overlap" of cells on each partition boundary. This overlap is visualized in figure 6.3. A spatial scheme operates gracefully using these overlapped cells when evaluated at an owned boundary cell. Specific to our implementation of a second-order spatial scheme, GEMS only utilizes one layer of ghost cells. The stencil order is maintained by communicating the gradients of border cells in addition to the state variables. This communication is shown in figure 6.4. Thus for our purposes, only one layer of interface cells is needed. Other solvers might have different communication paradigms.

The FOM algorithm can be expanded with realistic considerations as shown in algorithm 6. Within each sub-iteration of the Newton method, the boundary fluxes and gradients are communicated to adjacent processors as shown in blue. As a result, the convergence of the local sub-iterations represents a convergence of the global linear system. In an ideal case, this will lead to a computational speed-up of N/p as each processor has a fraction of the degrees of freedom. In reality, the communication between partitions

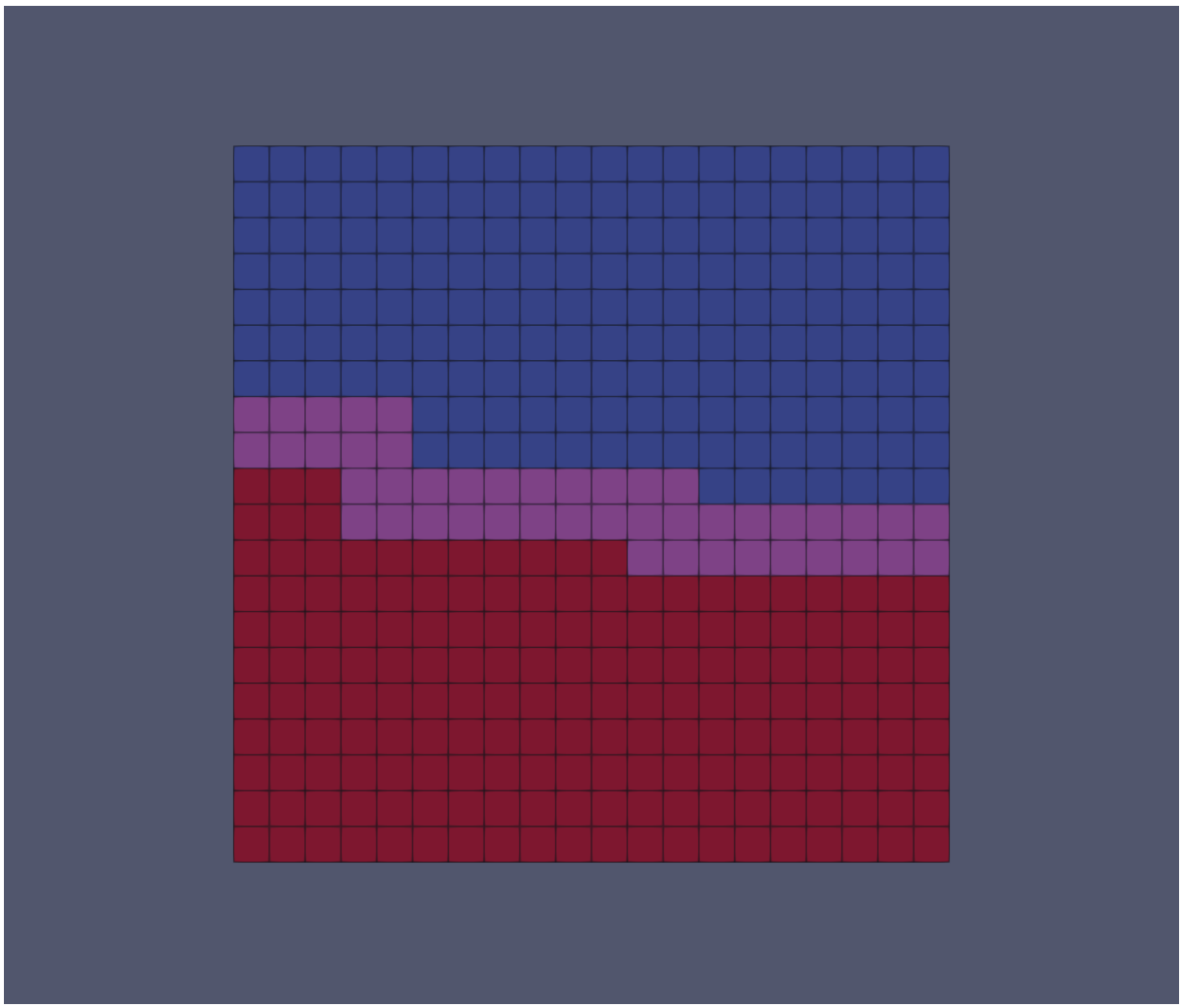


Figure 6.3: Example partitioning: each processor has one layer of partition overlapped cells.

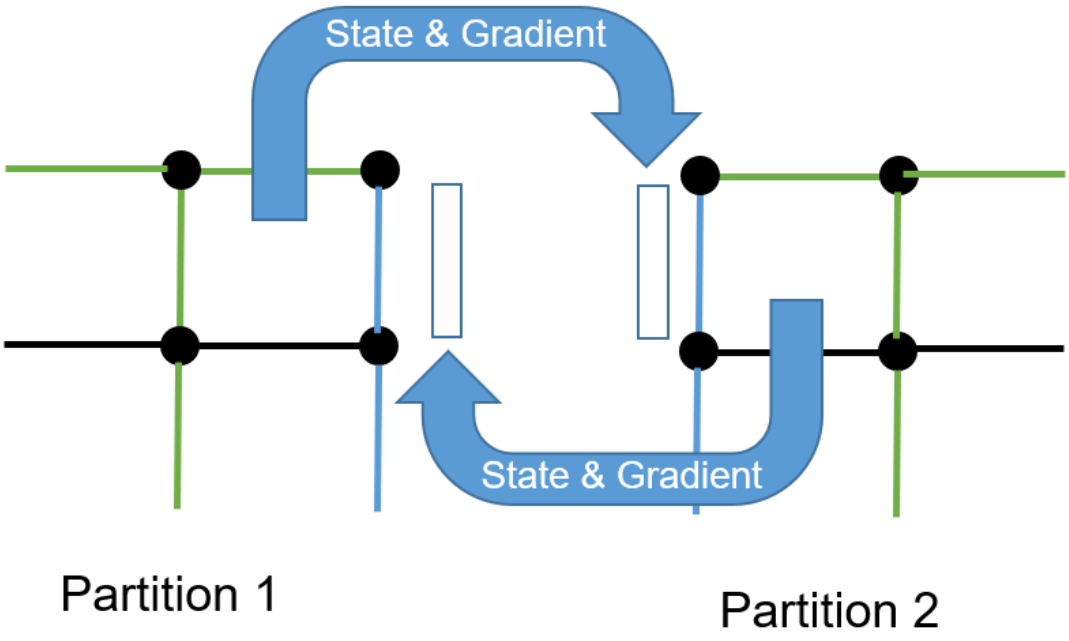


Figure 6.4: GEMS communication structure.

introduces a trade-off as increasing the number of partitions directly increases the cost of the communications. This is generally refered to in parallel computing as "overhead". The source of this overhead can come from a variety of sources from both a computer science (how the code was written) and hardware (how good is the silicon in your processor) perspective.

6.4 Computational bottleneck: Load Balancing

Recalling the 2D baseline case, figure 6.5 shows the efficiency of the baseline ROM for 2 and 44 processors. The efficiency metric here is compared to the FOM run on 2 or 44 processors, respectively. Both show appreciable speed-up; however, it is apparent that for lower sampling percentages, the speed-up achieved on 44 processors begins to saturate. There are improvements in efficiency as the sampling update iteration operates at the full dimension; however, the iterations evaluated on the sampled mesh will see diminishing returns.

Algorithm 6 FOM Solver Structure with Communication

Initialize Mesh $X \in \mathbb{R}^N$ and state $q \in \mathbb{R}^{N_{states}}$ collocated at each volume element
for $t \leftarrow t_0$ to t_M **do** ▷ Physical time-step loop
  for $i \leftarrow i_0$ to i_M **do** ▷ Sub-iteration loop
    for $c \leftarrow c_0$ to c_N **do** ▷ Loop over cell volumes
      $R(q) = \mathbf{f}(q, q^{t-1}, q^{t-2})$ ▷ Calculate cell residuals
    end for
    $\Delta q \leftarrow \mathbf{L}(R(q))$ ▷ Linear solve change in state from residual
    $q = q + \Delta q$ ▷ Update state for next sub-iteration
    Communicate state and gradient in partition halo
  end for
  $q_{n+2} = q_{n+1}$ ▷ Update previous time-steps
  $q_{n+1} = q_n$
end for

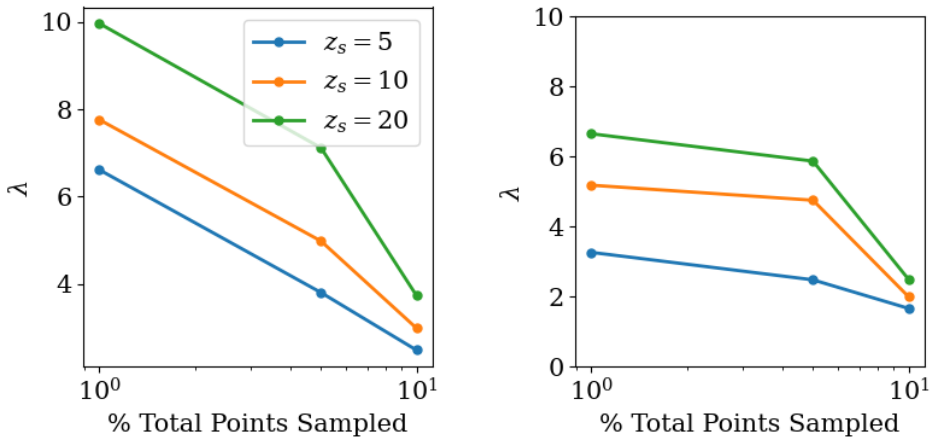


Figure 6.5: Baseline implementation efficiency on 2 processors (right) and 44 (left) processors for various sampling update frequencies and sampling percentages.

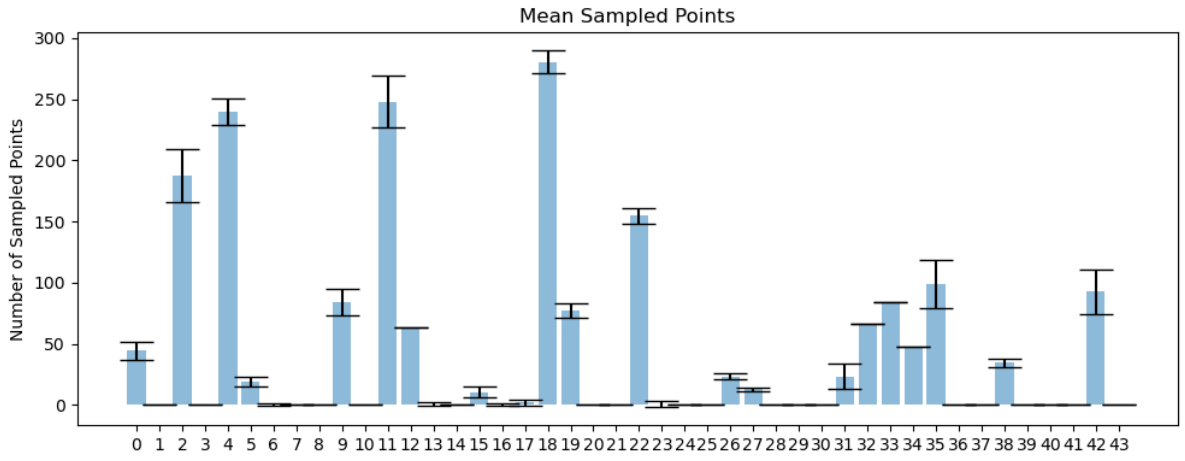


Figure 6.6: Visualization of relative processor load for 1D case 2D case on 44 processors.

It becomes clear that the baseline of the adaptive sampling methodology cannot scale to larger problems in two ways. The sampling algorithms do not consider the computational load balance in their point selection. More often than not, the region with the most sampling points is a relatively compact area in the overall computational mesh. The partition communication pattern is not updated to account for the disjoint nature of the sampled mesh. Mesh points that needed to communicate the full mesh partition boundaries no longer need to communicate during the sub-iterations operating on the sampled mesh.

The 44 processor case mean and standard deviation in the sampled mesh is shown in figure 6.6. Only 1/4 of the processors have any meaningful computational load and more than half have no work. Indeed under the worst-case scenarios 90% of available computational resources will be wasted waiting for over-encumbered processors. The computational mesh must be dynamically reallocated to the available computational resources to achieve meaningful acceleration. The following sections will describe the overall methodology and the context of the existing GEMS computational solver.

6.5 Load Balancing Framework

With the load imbalance visualized, we propose a framework to create a load-balanced implementation of the adaptive ROM. This framework is expanded in algorithm 7 with

additional steps in red and visualized in figure 6.7.

Algorithm 7 GEMS Adaptive Sampling Solver Structure

Initialize Mesh $X \in \mathbb{R}^N$ and state $q \in \mathbb{R}^{N_{states}}$ collocated at each volume element
Generate companion DMPlex full mesh
Calculate initial modal coefficients from initial basis **V** and sampling points **S**
From initial sampling points filter and distribute sampled mesh
for $t \leftarrow t_0$ to t_M **do** ▷ Physical time-step loop
  for $i \leftarrow i_0$ to i_M **do** ▷ Sub-iteration loop
    for $c \leftarrow c_0$ to c_N **do** ▷ Loop over cell volumes
      $R(q) = \mathbf{f}(q, q^{t-1}, q^{t-2})$ ▷ Calculate cell residuals if sampled
    end for
    $\Delta q \leftarrow \mathbf{L}(R(q))$ ▷ Linear solve change in sampled state from residual
    $q = q + \Delta q$ ▷ Update sampled state for next sub-iteration
  end for
  $q_{n+2} = q_{n+1}$ ▷ Update previous sampled time-steps
  $q_{n+1} = q_n$
end for
if mod(t,z_s) == 0 **then** ▷ Sampling and basis adaptation iterations
  Broadcast distributed sampled points back onto full mesh
  Approximate state at all points
  Update the basis at all points
  Calculate interpolation error at each point to select new sampling points
  From new sampling points filter and distribute sampled mesh
else
  Approximate state at sampled points
  Update the basis at sampled points
end if

To enable these additional steps, three core components need to be implemented.

1. **Translation:** With the host solver in mind, a method for converting the distributed mesh values to a format amenable to load balancing is required. This step is needed primarily to copy the data at the sampled points and prepare it for communication via a message-passing interface (MPI). This translation step is also needed to copy the distributed sampled mesh back to the original partitioned mesh for the full state basis and sampling update.

2. **Filtering:** Once the data is properly formatted, the identified sampling points are used to tag the required subset of points and deconstruct the unsampled geometric elements. This recovers a sub-mesh that only contains those sampled points. This mesh is still distributed according to the original partition scheme. This step is

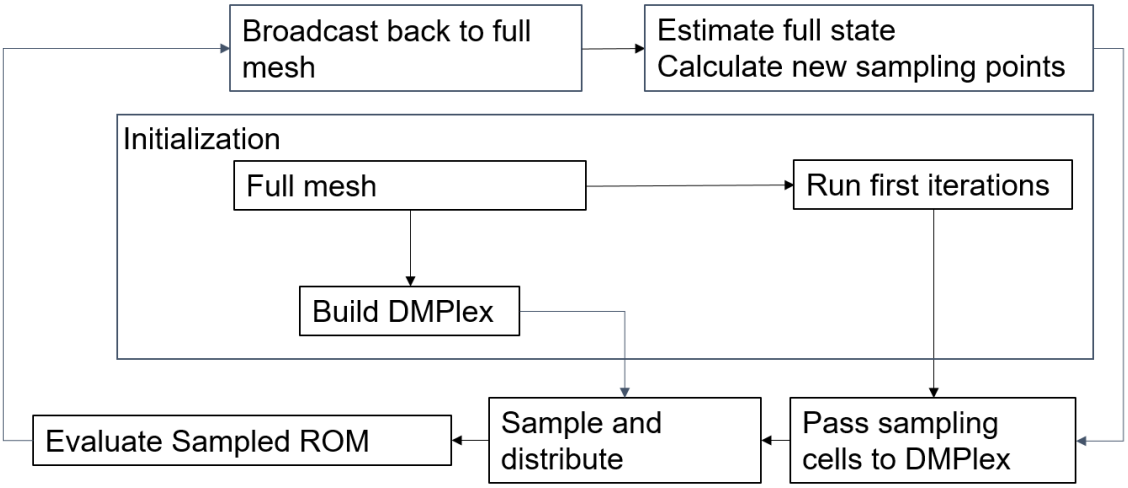


Figure 6.7: Visualization of load balanced ROM flow path.

where significant memory savings are achieved. In addition to deconstructing unused elements, a subpoint mapping is generated, which maps the unsampled mesh degrees of freedom to the sampled mesh. This mapping is used for data transmission between the entire mesh and sampled mesh and the transmission back to the original full mesh for estimation and resampling.

3. **Distribution:** After being filtered the mesh is still hosted according to the original partition layout. The distribution step takes those points and redistributes them across all the available processors. After the sampled mesh has been distributed a mapping between the filtered mesh and the new distribution is created. This mapping is used to scatter the degrees of freedom of each cell volume to their new host processor.

These three stages are visualized for a highly simplified mesh in figure 6.8. Here we can see the full mesh split into two equal partitions. A subset of points is selected, most of which fall into the lower partition. These points are filtered, and the remaining geometric elements are deconstructed. Finally, the sampled points are redistributed between the available processors.

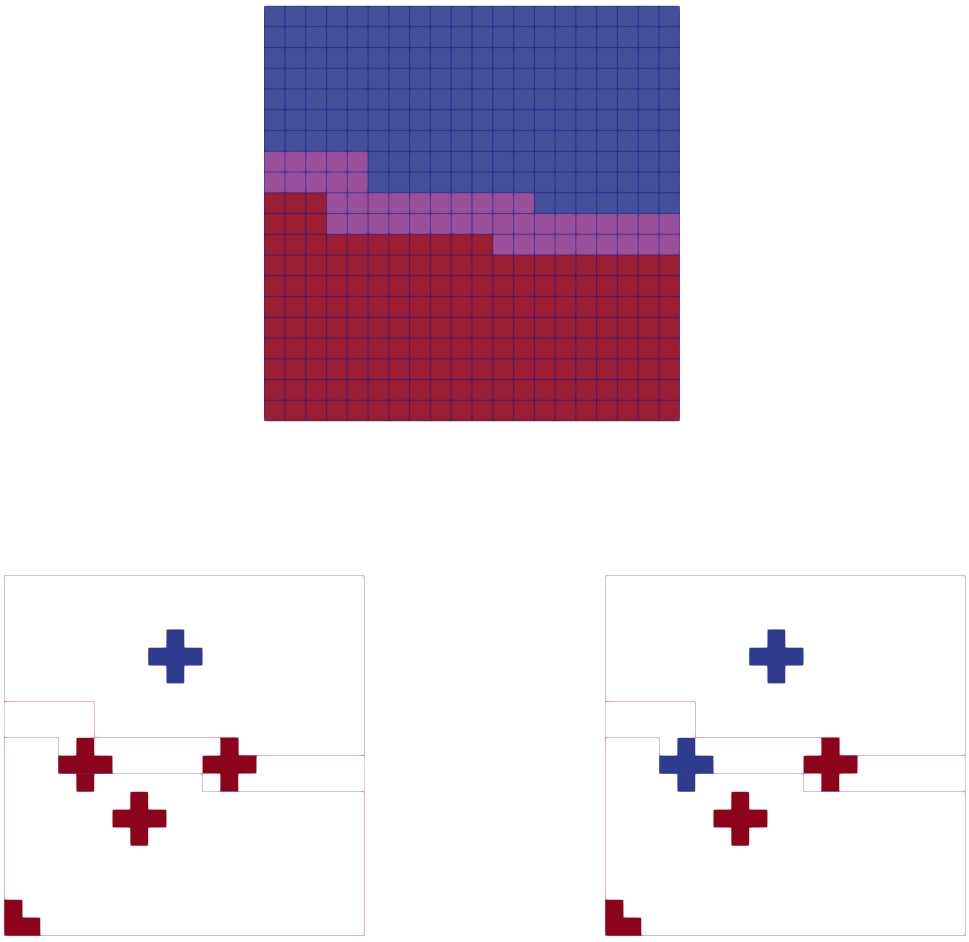


Figure 6.8: Simplified schematic of various components of redistribution: full DMPlex mesh (Top) filtered mesh (Bottom-Left) distributed mesh (Bottom-Right).

6.6 Implementation and Integration using PETSc

Fortunately, a wide array of available software can assist in our development of the load-balancing method. Load imbalance is something of great concern to adaptive mesh refinement (AMR) methods. In AMR methods, regions of high gradients are refined. A load imbalance can be created as processors that have refined significantly will have many more cells to evaluate than those that have not needed to refine or have been coarsened. The sampled ROM effectively does the opposite with the same result. The overall load balance is significantly compromised, with only a subset of the physical domain being evaluated.

As discussed earlier, the GEMS workflow was built in the context of a traditional computational fluid dynamics solver. In this scenario, the computational load does not change during the run time, so the partitioning can be static and fully computed offline. Directly leveraging this serial partitioner is insufficient and would require the full mesh geometry to be accessible to all the processors. This memory requirement is untenable and would completely offset any gains made by load balancing. As a result, the available libraries and software packages that are capable of performing in-situ reallocation of the mesh were inspected. Ultimately the Portable, Extensible Toolkit for Scientific Computation (PETSc) [93] was chosen. In addition to its more well-known linear and non-linear solvers, PETSc has various tools to enable and ease distributed computing methods. In particular, the `DM` mesh management system and, more specifically, the `DMPlex` unstructured mesh system was used. The overall methodology pursued is described in figure 6.8. A "companion" mesh is generated using PETSc objects during the GEMS solver initialization. The PETSc sub-mesh generation and distribution routines are then called to take the cells identified by the sampling algorithm and scatter them to all the available processors. While developed for use in AMR methods, these routines are general enough to be leveraged for acceleration via load balancing of the adaptive ROM.

6.6.1 PETSc Structures

PETSc is a highly comprehensive software package primarily used for its scalable linear solvers. [93, 94] However, it also contains extremely powerful utilities that simplify solver development on HPC. The following sections discuss the PETSc structures and strategies for achieving the sampling load balance. A majority of the effort of this section is associated with taking tools meant for AMR and "misusing" then in such a way that our ROM implementation is improved.

`DMPlex`

The `DMPlex` object is the general structure used to host a distributed mesh geometry. The `DMPlex` consists of a set of points on each processor. These "points" are not physical nodes but identifiers of physical discretization elements. These elements are what would normally be considered mesh components. Those are cells, faces, edges, nodes in three dimensions and cells, faces, and nodes in two dimensions. The connectivity of these elements forms a directed acrylic graph(DAG) which allows traversal of the different mesh elements. For example, consider a single 3D hex element. This element comprises 8 nodes, 12 edges, 6 faces, and 1 cell. A `DMPlex` object of this element would consist of 27 points, with point 1 corresponding to the cell, points 2-9 corresponding to the vertices, 10-15 corresponding to the faces, and finally, 16-27 corresponding to the faces. A schematic of this element is shown in figure 6.9. The order of points as cells, vertices, faces, and edges is not strictly required. However, it is the default of PETSc and makes it easier to discern a point type based on its value. In addition to this classification system, each type of point can be accessed by its constituent points (cones) and supporting (points). For example, in figure 6.9, the cone of cell one is composed of the six faces, and the cone of one of the faces is composed of its constituent edges and so forth. This works the opposite way with supports, with a face's support being its attached cells. Finally, we can traverse all of the connected points of a given point using its transitive closure, with a cell transitive closure being all the faces, edges, and points that constitute it. This allows relatively easy and intuitive traversal of mesh elements from a given point.

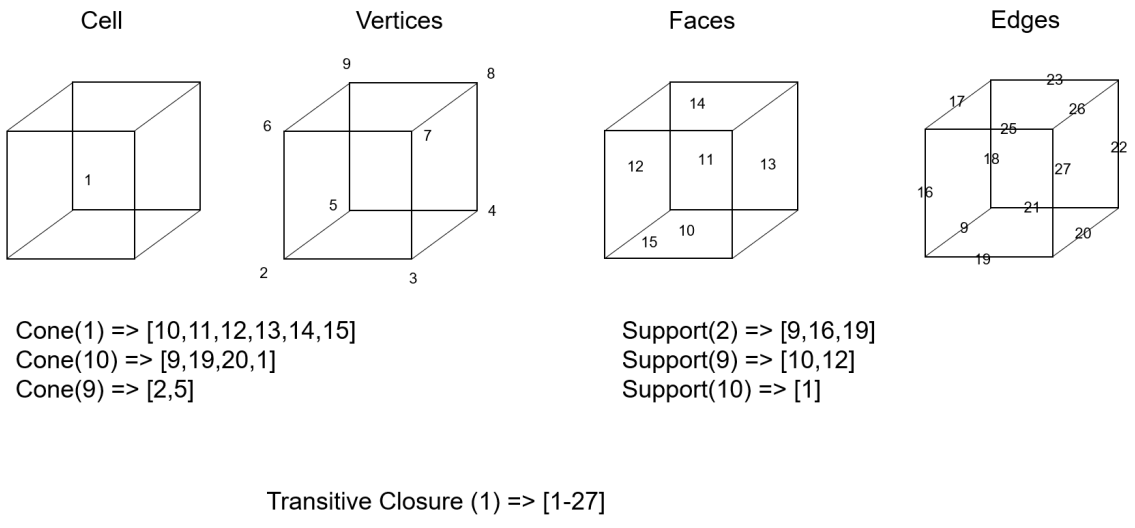


Figure 6.9: Schematic of DMPlex mesh.

The flexibility of the `DMPlex` object lies in how degrees of freedom (DOFs) can be assigned to any point. This means they can be used for a finite volume type method (with DOFs placed at cell centers and faces) or a finite element (DOFs placed at vertices). To be clear, assigning DOFs does not allocate the memory needed to contain them but instead informs the DMPlex where the DOFs will be located.

`Vec`

Accompanying the `DMPlex` is a Vec object storing the degrees of freedom. At any point, a Vec object can be queried from the `DMPlex`. This vector will be a contiguous block of memory containing all the DOFs corresponding to those associated with points in the `DMPlex`. In the context of a fluid dynamics solver, the density, momentum, energy, and scalar states are stored contiguously for each physical point. This is a significant advantage over GEMS where each cell is an independent defined structure as described above.

`PetscSF`

The final major component of PETSc we are leveraging to implement load balancing is the PetscSF (star forest). This structure generically describes to the communication of information between distributions of points. For our purposes, this is used in two ways.

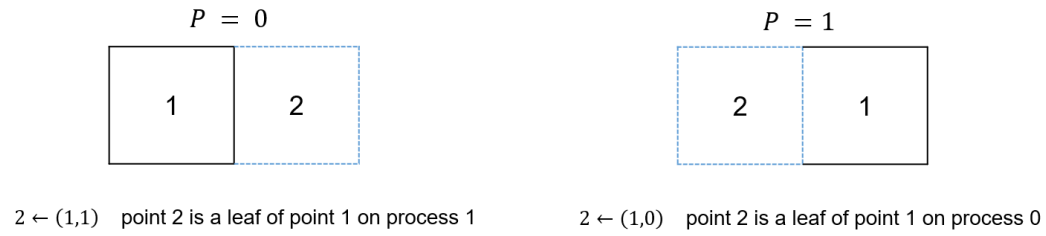


Figure 6.10: Example schematic "overlap" star forest.

1. The "overlap" layers are needed to reconstitute the GEMS mesh structure. The `DMPlex` objects described above can be considered independent mesh, each on a different process. The SF for this is a companion to the `DMPlex` that describes the partitioned cells. This structure is visualized in figure 6.10. This star forest is designed to communicate the partitioned overlap cells, but for integration with GEMS it is used to build the communication layout. In the context of the star forest, a point is either a root or a leaf. Every point is considered to only be "owned" on a single processor. The point is considered the root, and all other copies are leaves of that point.

2. The "distribution" star forest is significantly different. This star forest is created when PETSc redistributes an existing `DMPlex` object. The star forest, in this case, describes every processor of where their point will end up if they are being rehosted on a different processor. This is visualized in figure 6.11. In the load balancing implementation context, we use this distributed SF to broadcast the state and basis information to the destination processor and then copy the state and basis information back to the original partition to execute the sampling update.

These three structures, `DMPlex`,`Vec`, and `PetscSF` are used to achieve the load balancing between sampling iterations.

Recalling algorithm 7 we use the PETSc objects to complete each of these steps. At the code initialization, a "companion" `DMPlex` is generated that directly maps onto the GEMS mesh format. Once the first sampling adaptation step occurs, the sampled cells on the FOM solver side are copied into the companion `DMPlex` object. At this point, the companion mesh is identical to the full-order partitioned mesh. Following

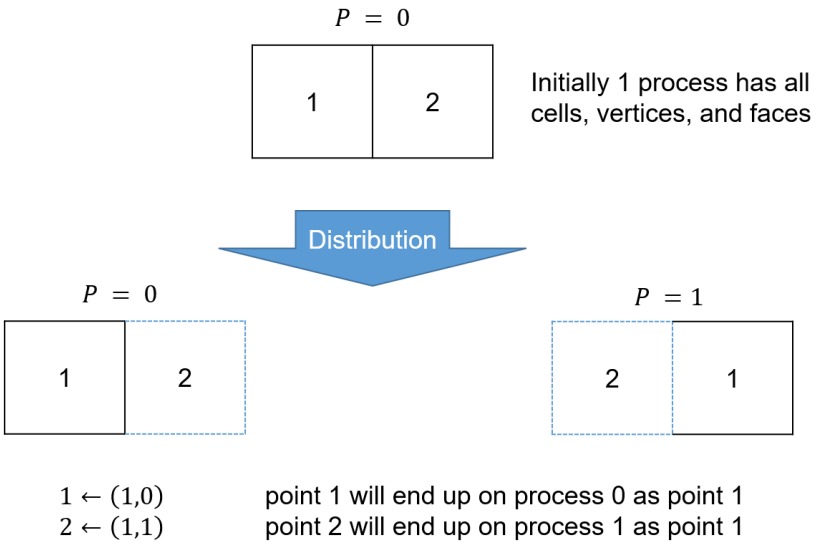


Figure 6.11: Example schematic of the partition star forest.

this, the `DMPlex` is filtered to deconstruct unsampled elements and then distributed to all available processors. From this point on, there is no guarantee of where any physical point is located within the processor set. Using the distributed `DMPlex` we now generate an equivalent sampled mesh format and copy the basis and state. The ROM then iterates on this sampled mesh format until the next sampling update iteration. At this point, the state and basis of the sampled mesh are passed back up the chain, from the sampled mesh to the distributed `DMPlex` to the full `DMPlex` and back to the full partitioned mesh. After this broadcast back to the full mesh the sampled mesh and corresponding DMPlex objects are deallocated. The sampling update full FOM iteration then occurs and a new set of sampling points is generated and the cycle repeats.

6.7 Results

Returning to the previous baseline cases with the newly established framework to examine the achieved efficiency.

6.7.1 1D laminar flame

We start with the 1D laminar flame propagation case. A priori, we do not expect this case to show significant speed-up with the addition of load balancing. The 1D case FOM only has 1000 mesh elements and is partitioned into 2 parts. Therefore, each processor will have negligible work or waiting time even during the adaptive ROM. Under these circumstances, we don't expect a significant improvement, making a poor test bed for testing scalability. However, it makes an ideal case for verifying load-balancing implementation, particularly in verifying the intricacies of regenerating a GEMS-compatible mesh structure.

With the framework in place, the components of the code to be timed are:

1. **Calculation Time:** These are local computations on the MPI processor. The primary contributors are the evaluation of the residual, jacobian, and linear solve. This component is where we expect to see the computational savings from load balancing, as the cost of these operations is directly proportional to the local cell count.

2. **MPI Communications:** In a MPI paradigm, communications refer to the messages sent between different processors. Referring back to figure 6.4 this is the time spent sending and receiving adjacent partition cells' flux and gradient components. The act of redistributing the sampled mesh gives the partitioner the ability to create disjoint graphs with no communication overhead. As a result, we expect to see some improvement. However, because communications are a relatively small portion ($< 25\%$) of the runtime, the savings are not expected to be significant.

3. **I/O:** These refer to time spent writing fields or monitoring probes to disk. This can be significant for bigger cases, but the load balancing is not expected to improve this, and they are included for completeness.

4. **Repartitioning:** This timing is only relevant for the load-balanced ROM. This includes the time spent filtering and redistributing the sampled mesh and the time

needed to send the states and relevant cell quantities to their new host processor. The method's viability is made here as any savings in the previous components may be offset by the cost of the redistribution.

As with the baseline, the original efficiency metric used

$$\lambda = \frac{t_{FOM}}{t_{ROM}}. \tag{6.2}$$

An additional metric measuring the relative performance of the load-balanced ROM compared to the baseline ROM is introduced

$$\lambda_{LB,ROM} = \frac{t_{ROM}}{t_{ROM_{LB}}}, \tag{6.3}$$

and overall performance of the load-balanced ROM vs. the FOM

$$\lambda_{LB} = \frac{t_{FOM}}{t_{ROM}} \times \frac{t_{ROM}}{t_{ROM_{LB}}} = \frac{t_{FOM}}{t_{ROM_{LB}}} \tag{6.4}$$

With these timing components in mind, we examine the breakdown for the 1D laminar flame case (figure 6.12). Starting the comparison to the FOM, both the load-balanced ROM and FOM show significant speed up with $\lambda = 5.8$. The load-balanced ROM shows minor improvement with a $\lambda_{LB,ROM} = 1.02$. These combine for a combined efficiency of $\lambda_{LB} = 6.0$. The breakdown shows that the calculation and MPI cost for the load-balanced ROM are reduced compared with the baseline ROM. However, these are almost completely balanced by the time spent redistributing the sampled mesh.

This is not unexpected. Recall that the 1D flame case already struggles to provide a significant computational load, as approximately 20 cells remain after sampling.

For higher values of z_s (the frequency of sampling updates), the overall performance of the load-balanced ROM improves significantly. For $z_s = 10$ and $z_s = 20$ a improvement of $\lambda_{LB,ROM} = 1.3$ and $\lambda_{LB,ROM} = 1.5$ is observed, respectively. This outcome is consistent with expectations as the primary cost associated with load balancing is the repartitioning step. In all other components, an improvement in wall time is expected.

Finally, it should be noted that while a load balancing improvement of 1.5 over the

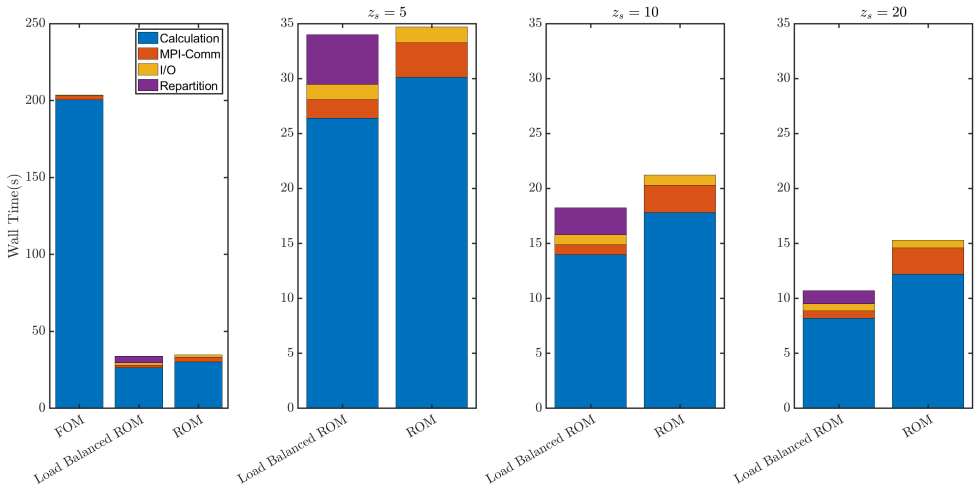


Figure 6.12: Timing breakdown of 1D Flame Case.

baseline ROM is a relatively minor improvement, it is multiplicative with the efficiency λ. This leads to a combined efficiency over the FOM of $\lambda_{LB} = 11.2$ and $\lambda_{LB} = 19.0$ respectively.

With these observations in mind, the intent is now to apply the load-balanced ROM to significantly larger problems where greater performance may be possible as we increase the processor count.

6.7.2 2D Rocket Injector Load Balanced Performance

The load-balanced adaptive ROM is first compared with the baseline ROM the verify consistency between the two implementations. The point monitor as identified in figure 5.10 is shown comparing the two implementations with a 1% sampling rate with a sampling adaptation interval (z_s) of 5. The two match exactly and are accurate within floating point precision. A small amount of variation due to the non-associative property of floating point addition is observed. Because the mesh points are distributed to various processors the summation of the face fluxes is not guaranteed to be computed in the same order.

To more clearly visualize this change of ownership, the feature outline of the full mesh partitions is overlayed with the sampled mesh host process in figure 6.15.

Compared with the 1D example, the rocket injector has significantly more complex

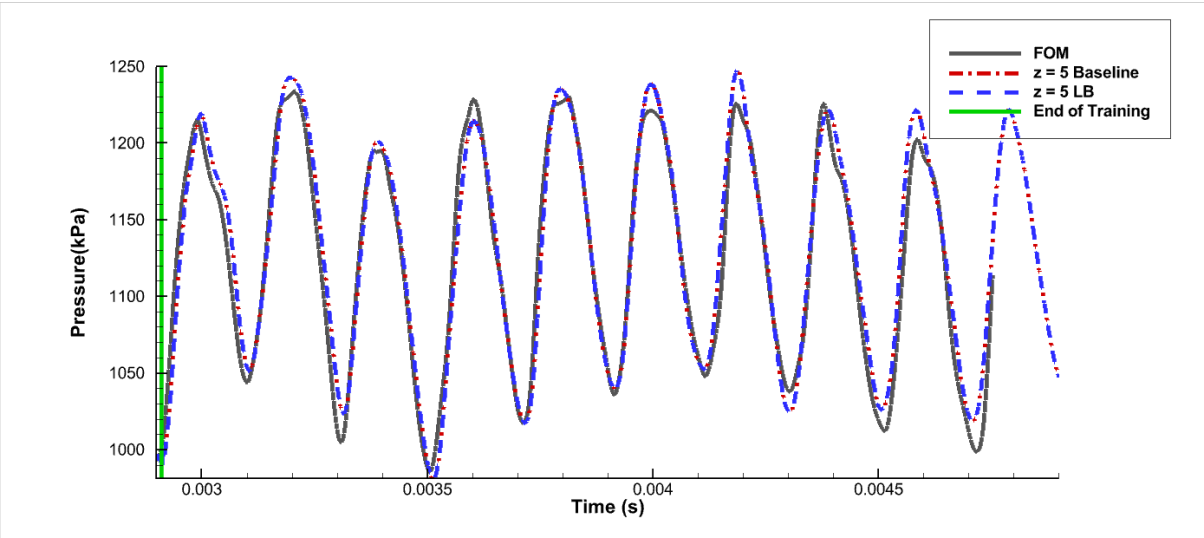


Figure 6.13: Pressure signal comparison of FOM and ROM baseline and load balanced implementation. Training window: $t = 0.0029 - 0.00291$s.

geometry and resulting dynamics. The load balancing effect is visualized in figure 6.14. Here temperature contours show the dynamic evolution of the flow field. Overlayed on the whole mesh is sampled mesh colored by their host rank. This sampled mesh is observed to change over time as expected, and additionally, we can see the various sampling "blobs" switch from process 0 to process 1.

With the results validated against the baseline ROM and the expected distribution behavior observed, the resulting performance is examined. The performances are quantified in figure 6.16 broken down into component wall times.

First, comparing the performance of the FOM to the baseline ROM, a $\lambda = 7.13$ is achieved for a $z_s = 5$. However, when comparing the load-balanced ROM with the baseline ROM, we see a $\lambda_{LB,ROM}$ of 1.6. This, combined with the ROM improvement, leads to combined observed efficiency of $\lambda_{LB} = 11.8$ when comparing the load-balanced ROM with the FOM. We can observe that the compute portion of the wall time is significantly reduced for the load-balanced ROM. Unusually, the relative efficiency between the baseline and load-balanced ROM decreases with an observed $\lambda_{LB,ROM}$ of 1.32 and 1.5 for a respective z_s of 10 and 20. Compared with the original FOM these efficiencies correspond to a combined speedup λ_{LB} of 13.1 and 19.54 respectively. This is due to the relative load balance of the sampled points before redistribution. For the current load-balanced

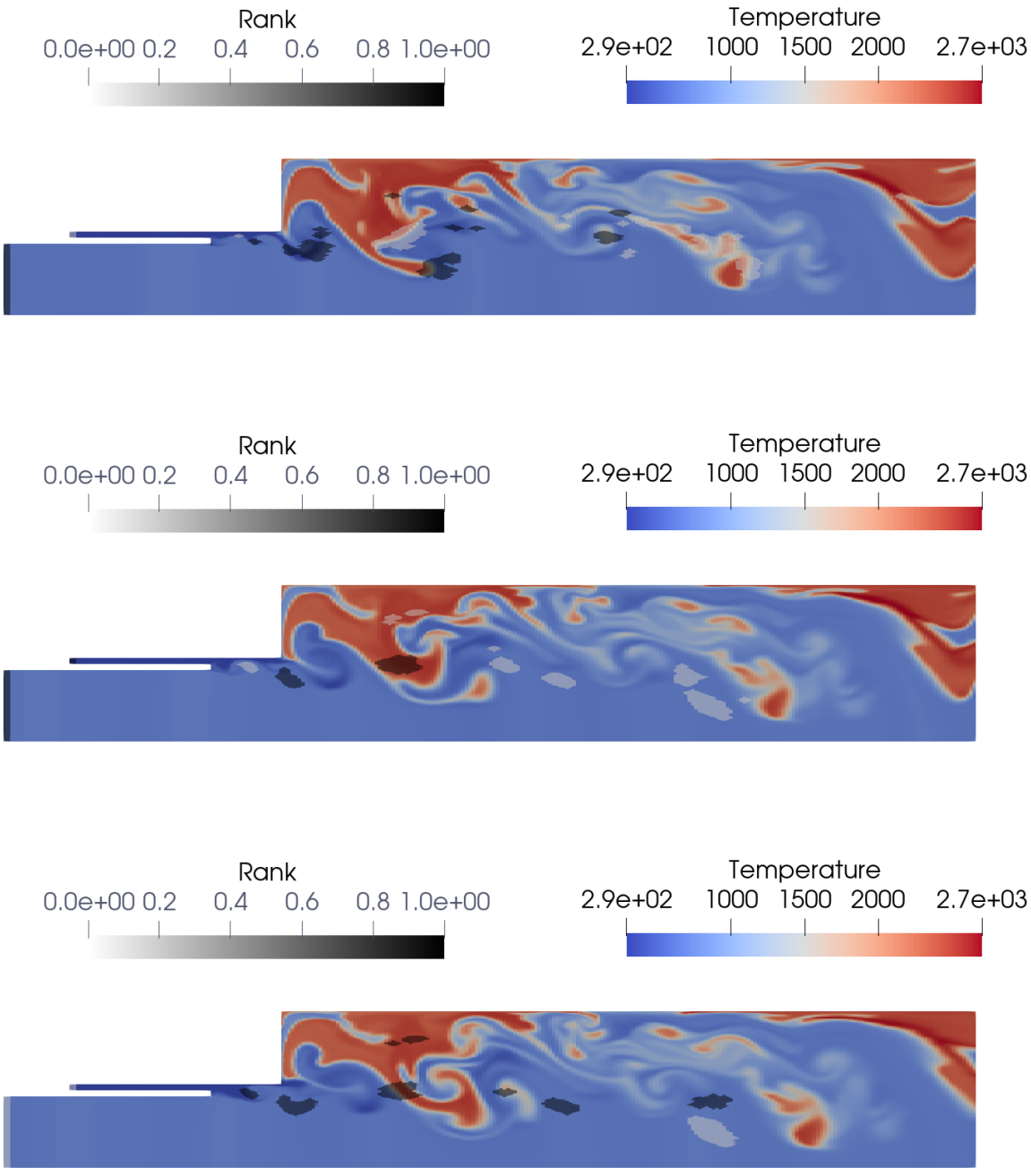


Figure 6.14: Dynamic evolution of 2D rocket injector: Contours of temperature overlayed with sampled ROM points colored by MPI process.

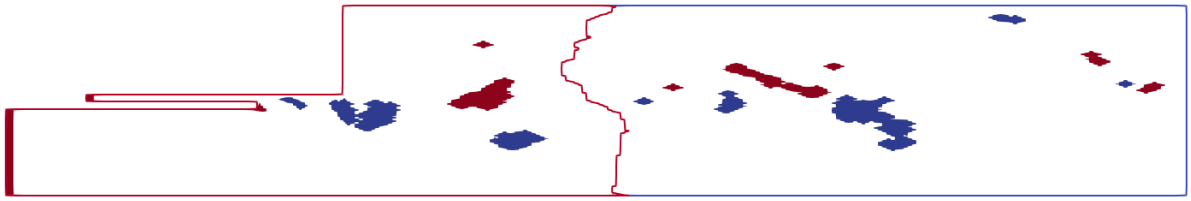


Figure 6.15: Visualization of the change in ownership of sampled cells on two processes for a 2D rocket injector for a single sampling set.

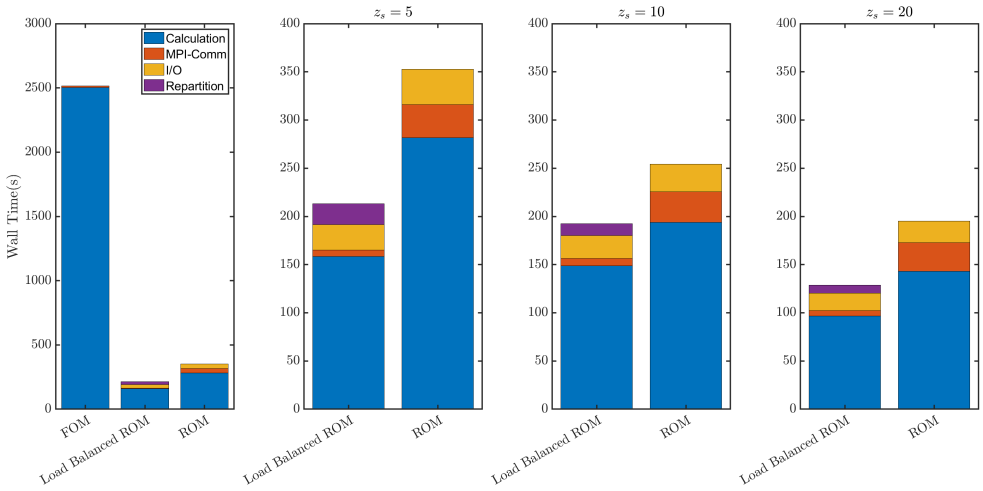


Figure 6.16: Timing breakdown of 2D rocket injector case.

ROM implementation, the sampled cells will always be redistributed regardless of their host processor. One would expect for the worst-case scenarios if the sampling update produces a perfectly balanced sampling set, the load balancing would only hurt the performance. The change in sampling update for $z_s = 10$ and $z_s = 20$ is observed to change the initial balance of points compared with $z_s = 5$, reducing the effectiveness of the load balance. Ultimately, the load balancing improves wall time in every test case, even if it does not provide a significant improvement.

6.7.3 Current Limitations & Proposed Improvements

Despite the performance considerations observed in the test cases, the load-balanced implementation of the ROM is critical for reaching the theoretical efficiencies promised by projection-based ROMs. The current implementation of load balancing suffers from several deficiencies. Extending this method for extremely large 3D problems while not a

theoretical challenge requires a significant time investment to develop and test.

First, as a result of the GEMS solver idiosyncrasies, the memory requirement for the solver, when operating in this way, is effectively double, with two copies of the entire mesh and two copies of the sampled mesh being allocated at the same time. The load balancing also continuously translates the mesh format between PETSc and the GEMS solver. A significant overhead of the re-partitioning would be removed by having the solver operate directly on the PETSc mesh data structure. This would also cut the memory requirement of the method in half, which is critical for deployment on smaller personal computers.

The second major shortcoming is a deviation observed during the sampling adaptation step. This is relatively minor in small cases but has been causing significant issues in application to larger problems. The reason for the deviations is believed to be a result of the load-balanced sampled mesh. The time integrator logic is visualized in figure 6.17. If a previously unsampled cell becomes sampled during the update step during the first ROM sub-iteration, the 2nd order time integrator will attempt to access the n-2 timestep for that sampled cell. However, because the cell did not exist in memory during the previous sampling cycle, the time integration cannot access it. This effect causes the first sampled iteration after the sampling update to degrade to first-order accuracy in time. Resolving this issue is non-trivial as it would require a priori knowledge of the sampling update cells.

6.8 Summary

This chapter together with chapter 3 are primarily focused on the computational tractability of projection ROM methods. While chapter 3 focused on the offline cost of ROM preprocessing this chapter has focused on overcoming the challenges associated with adaptive sampling methods. These methods are critical to achieving practical use in ROM but introduce significant computation challenges. By applying tools developed for MPI scale adaptive mesh refinement we are able to improve the efficiency of the adaptive ROMs significantly. The gains are dependent on the sampling clustering but are generally any-

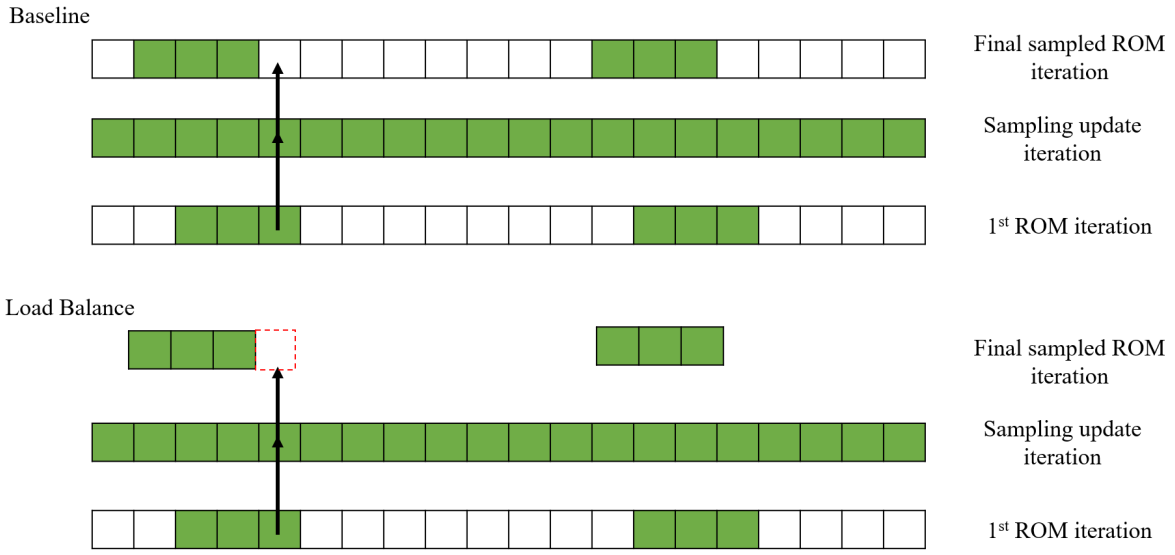


Figure 6.17: Visualization of the degraded time integration scheme.

where from 1.1 to 5 times faster. While this speed-up is fractional, when combined with the inherent efficiency gains from hyper-reduction these methods are required to realize the true potential of adaptive ROMs which can be 2 to 3 orders of magnitude faster.

Chapter 7

Summary and Conclusions

7.1 Contributions

This thesis investigated the development and application of projection-based reduced-order models to multi-scale transport problems. Particular emphasis was given to the grand challenge of providing accurate predictive fidelity at a fraction of the computational cost of full-order models. Projection-based ROMs have traditionally seen limited application on significant scale and complexity flows. Historically, this family of ROMs has been of limited utility in predicting convection-dominated and multi-scale problems due to the limitations of linear projection. This thesis contribution demonstrates a significant improvement in the predictive capabilities of ROMs for problems far beyond the scale of those found in the literature. Here the conclusions of the various contributions are highlighted.

Chapter 2 provides the computational basis for the contributions of this work. The Large-Eddy Simulation (LES) methods used as the full-order model are presented. The various turbulence and reacting flow models are discussed in detail. Following this, the theoretical basis for projection-reduced order models is presented. Standard formulations, most notably Galerkin and least-squares Petrov Galerkin, are discussed. The development of the newer MP-LSVT ROM formulation is discussed to significantly improve the local and global stability of ROMs of reacting flows. Finally, sampling methods critical to the expected computational efficiency of ROMS are introduced.

Chapter 3 starts by describing the computational challenges in the required prepro-

cessing for projection-based ROMs, particularly the memory and I/O challenge for large cases. Parallel Linear Algebra Tool FOr Reduced Modeling (PLATFORM) was developed to address these memory and I/O challenges. PLATFORM leverages the ScaLAPACK package with various I/O techniques to significantly speed up the required pre-processing. Offline computations that could require days to complete are reduced to minutes. Because of the similarity in ROM preprocessing and modal decomposition techniques, the tool provides similar efficiency gains for using general modal analysis. This efficiency is leveraged on the large data sets of the two GTMCs studied to perform DMD analysis and the required pre-processing for the companion ROM studies. In addition to enabling work by various collaborators, this package has been made available to - and used by - the ROM community in the form of an open-source repository.

Chapter 4 exhibits a large-eddy simulation (LES) of a methane-air dual-swirl gas turbine model combustor using a flamelet-based approach. The simulation is validated with experimental measurements. Averaged features, such as the recirculation bubble and overall flame shape, are well captured. In addition, unsteady probe measurements show a good correlation between unsteady effects in terms of power spectral densities. The unsteady analysis is supplemented with dynamic mode decomposition comparison to the experimental and computational results. The Flame A simulation, in particular, shows a high degree of accuracy compared to experimental data and is used to develop ROMs.

This full-order model is used as training and testing data for developing large-scale ROMs. The MP-LSVT approach builds upon previous methods to significantly improve the robustness of the ROM. This method improves global stability by minimizing the discrete residual, guaranteeing symmetrization. This is augmented by using local limiters to mitigate non-physical artifacts. These combine to significantly improve the ROMs ability to reconstruct dynamics within the training region with L2 errors of less than 5%. However, due to the chaotic nature of turbulent combustion in the GTMC system, a static basis approach is ill-suited for the predictive modeling of these problems. This limitation is quantified via computation and visualization of the information lost due to

the static projection. Attempting to use a static basis method beyond the training region will result in error spiking to values greater than 50 % and coherent structures becoming non-physical.

An adaptive basis method is used, which leverages information from the full-order model operator to update the basis and minimize this projection error. Using this method, the ROM can predict the combustor dynamics far outside the training region and shows improvement over static basis methods. In addition, this significantly reduces the ROM training requirements, where the equivalent static basis ROMs require thousands of snapshots. This represents a reduction in training by a factor of 100-1000. This performance comes at the cost of requiring evaluation of the FOM operator. This leads to a computational efficiency that is slower than the original FOM. Future efforts on this scale of the problem will leverage the hyper-reduction to maintain the predictive accuracy of the ROM while improving computational efficiency.

The major source of computational efficiency gains in projection-type ROMs is hyper-reduction-based sampling. In chapter 5 an adaptive sampling method is presented to link the predictive capability of adaptive basis methods with the efficiency gains of hyper-reduction. The capabilities of this method are exhibited and quantified for a 1D laminar flame and 2D rocket injector case. Adaptive sampling shows a high degree of accuracy (less than 5% L2 error) and significant computational efficiency (3 - 10 times faster). In addition to purely predictive ROM usage, parametric uses of the ROM are conducted with significant improvement(less than 20% error) compared with static basis methods. Finally, use of adaptive ROMs as accelerators of traditional LES workflows is exhibited. The ROMs show the capability to accelerate numerical transients by orders of magnitude to produce a more physical initial condition.

The computational bottleneck associated with the in-situ adaptation of sampling points limits the ability to maintain efficiency as we increase the number of processors. In chapter 6, an in-situ distribution methodology is developed to load balance the sampled points across available processors to address this shortcoming. Higher levels of efficiency are achieved by the in-situ balancing of computational load while simultaneously reducing

inter-process communication. This improvement comes at the cost of overhead associated with redistributing sampled points. The improvement of this method is analyzed on the baseline adaptive sampling cases. Overall across all cases, an improvement in efficiency is observed, ranging from 2 times to 5 times. The implementation efficiency combines with the overall ROM method to produce speed-up of 2 to 3 orders of magnitude relative to the FOM. This improvement is problem dependent and directly proportional to the relative imbalance of the initially selected sampling points. Ultimately, the implementation strategy at the HPC scale is just as critical as the ROM formulation to achieve significant computational speedup.

This dissertation takes valuable strides in making ROMs more palatable for practical problems. This is primarily done by synthesizing a variety of newly developed ROM methods. The originality of the work is by taking these resulting frameworks and overcoming a variety of computational problems associated with how the ROM methods interact with traditional solver frameworks. Both online and offline aspects of these problems are addressed and accelerated by as many as three orders of magnitude.

7.2 The Future of ROM-based Computational Solvers

This chapter, combined with chapter 3 places significant focus on developing and exploring the scientific computing techniques needed to apply projection-based reduced-order models to practical problems. Given the expertise gained during the development of the ROMs, several recommendations for developing future ROM solvers for physical systems can be made.

7.2.1 Pre-Processing

A major hurdle in this thesis was building a framework that supports taking FOM solver outputs and processing them into ROM solver inputs. The data format, in particular, can be extremely painful to navigate. Using a standard data format (e.g., HDF5) for input and output fields is highly advised, saving significant development time and making

visualization of ROM quantities much easier. Leveraging in-situ processing (e.g., streamingSVD) to the FOM solver significantly accelerated the development process. A mature ROM solver should not require difficult pre-processing but instead, gracefully transition from an initial FOM iteration to ROMs built from a few initial FOM time snapshots. This structure would make ROM solvers functionally the same as FOM solvers from a user standpoint.

7.2.2 Data locality

Spatially discretized codes commonly use abstraction structures that contain local variables. For example, a "cell" object may have velocity, density, temperature, etc., as components. This structure is common as it significantly eases the readability of code and makes intuitive sense. FOM solvers then construct a sparse linear system from these abstractions to advance the system. ROMs, conversely, are predicated on dense linear algebra, and using this structure can reduce efficiency via the construction of the dense system. Using a mesh data system like that described in Chapter 5, where discretized data is stored contiguously, greatly alleviates computational considerations from reconstructing linear systems. For clarity, consider the construction of the state vector. With cell structures, we must first allocate the full vector and then copy the contents of each cell to the vector. In contiguous data storage, the data is pre-arranged, and the cells are identified via strides.

7.2.3 Sampling Redistribution

In this work, we used PETSc to redistribute the sampled ROM points. Fundamentally we are using a tool designed for adaptive mesh refinement to serve our slightly different purpose. A flaw in this is that the redistribution does not handle small voids gracefully. If there are small gaps (1-2 cell widths) between sampling points, the partitioner cannot distinguish if it needs to facilitate communication across these voids. A method to determine this communication is required based on the sampling point, and informing the partitioner would be quite valuable. The communication reduction of disjoint graphs is

not to be underestimated, especially as it scales to larger problems.

7.2.4 Trajectory of ROM Solvers

With the starting point of existing mature full-order model solvers for various physics, we propose a trajectory for ROM usage/development.

1. **ROM modules:** Ultimately, projection-based ROMs still require the evaluation of the full-order model operator (at a few sampling locations). As a result, testing ROM solvers will be developed via modular attachment to existing physics-based solvers. These are primarily used to evaluate ROM performance on new physics.

2. **Hybrid Solvers:** As discussed in this thesis, integrating the two solver paradigms becomes critical as the focus of ROM development shifts from accuracy to efficiency. Solvers developed with this mindset will be the first that can realize the efficiency of ROMs while operating in a similar framework as FOMs.

    - **ROM methods in the context of heterogeneous architecture:** Modern high-performance computing has pivoted heavily into hardware acceleration capabilities. Communication on these systems is the primary bottleneck. ROMs like those explored in this thesis can, in some ways, alleviate this, as the disjoint graphs produced by hyper-reduction are extremely efficient. However, ROM methods also have global coefficients accessed by all degrees of freedom, negatively affecting computational gains. It will remain to be seen how these factors compete.

3. **ROM primary users:** As these efficient ROM solvers become available, they will likely take an important role in the design cycle. The major limiting factor to this aspect is the uncertainty quantification of the produced results. Unlike physics-based reduced-order models, there is limited physical intuition into the errors that arise from ROM. A new generation of users will have to be able to balance an understanding of the physical systems with the computational realities

of projection-based ROMs. The intricacies of ROM use would ideally take the same form as current CFD engineers' understanding of CFD solvers (e.g., spatial discretization error, model-form deficiencies).

7.3 Avenues for future work

Data-driven reduced-order modeling of multi-scale transport problems of significant scale is a grand challenge problem. Essential strides are made across various areas in this thesis, but many challenges remain. Below are a variety of future avenues that the author feels are critical in developing ROMs to reach the general user.

- **FOM/ROM Combined Solver:** Most ROM solvers are constructed as add-ons to existing FOM solvers. This is understandable, as most ROMs are developed to accurate existing FOM solvers. However, with the maturation of ROM methods, implementation is increasingly becoming the critical bottleneck. Because most physical systems can be discretized using a sparse linear system, computational solvers are designed around this. ROM methods require significant dense linear algebra leading to unusual trade-offs within ROM modules built into existing FOM solvers. A solver designed from the ground up to facilitate both FOM and ROM paradigms would be an extremely useful tool to develop ROMs and reach the theoretical efficiencies promised by their formulations.

- **Operational Studies of ROM solvers:** The ultimate goal of ROM methods is to allow access to the predictive capabilities of FOMs at a fraction of the computational investment. However, ROMs can also be used as a companion to aid in FOM analysis. An example of this is a ROM that could be used to accelerate the collection of statistical data or perform a set of parametric studies with a FOM, then being used to investigate those of the highest interest.

- **Mesh Reliance:** While providing significant online cost reduction, the ROMs described in this thesis require the same preparation to initialize as the FOM. The

geometry of the problem of interest is discretized with a mesh, and ultimately, the ROM methods are just as dependent on this initial step as the FOM. The FOM is treated as the ground truth in this thesis, but ultimately, even a perfect ROM would mimic any deficiencies present in the FOM. In the many-query application, this preparation work is often automated. It would be interesting to see how these ROMs perform on autogenerated or dynamically adapted mesh systems. This would also make ROM methods more mature for practical use within an industry solver, which often uses an autogenerated mesh.

Bibliography

[1] S. Nalley and A LaRose. Annual Energy Outlook 2022. Technical report, Energy Information Agency, 2022.

[2] Sébastien Candel. Combustion dynamics and control: Progress and challenges. Proceedings of the Combustion Institute, 29(1):1–28, January 2002.

[3] P. Weigand, W. Meier, X. R. Duan, R. Giezendanner-Thoben, and U. Meier. Laser Diagnostic Study of the Mechanism of a Periodic Combustion Instability in a Gas Turbine Model Combustor. Flow, Turbulence and Combustion, 75(1-4):275–292, December 2005.

[4] M. Stöhr, Z. Yin, and W. Meier. Interaction between velocity fluctuations and equivalence ratio fluctuations during thermoacoustic oscillations in a partially premixed swirl combustor. Proceedings of the Combustion Institute, 36(3):3907–3915, January 2017.

[5] Christoph M. Arndt, Michael Severin, Claudiu Dem, Michael Stöhr, Adam M. Steinberg, and Wolfgang Meier. Experimental analysis of thermo-acoustic instabilities in a generic gas turbine combustor by phase-correlated PIV, chemiluminescence, and laser Raman scattering measurements. Experiments in Fluids, 56(4):69, March 2015.

[6] P. Weigand, W. Meier, X. R. Duan, W. Stricker, and M. Aigner. Investigations of swirl flames in a gas turbine model combustor: I. Flow field, structures, temperature, and species distributions. Combustion and Flame, 144(1):205–224, January 2006.

[7] Mallard E. Recherches Experimentales et Theoriques sur la Combustion des Melanges Gaseux Explosifs. Ann. Mines, 8(4):274–568, 1883.

[8] John William Strutt Baron Rayleigh. The Theory of Sound. Macmillan & Company, 1896. Google-Books-ID: A7fvAAAAMAAJ.

[9] F. E. C. Culick. Combustion Instabilities in Propulsion Systems. In F. Culick, M. V. Heitor, and J. H. Whitelaw, editors, Unsteady Combustion, NATO ASI Series, pages 173–241. Springer Netherlands, Dordrecht, 1996.

[10] F. E. C. Culick. A Note on Rayleigh's Criterion. Combustion Science and Technology, 56(4-6):159–166, 1987.

[11] Boa-Teh Chu and S. J. Ying. Thermally Driven Nonlinear Oscillations in a Pipe with Traveling Shock Waves. Physics of Fluids, 6(11):1625, 1963.

[12] Sebastien Ducruix, Thierry Schuller, Daniel Durox, and Sebastien Candel. Combustion Dynamics and Instabilities: Elementary Coupling and Driving Mechanisms. Journal of Propulsion and Power, 19(5):722–734, 2003.

[13] Geo A. Richards, Douglas L. Straub, and Edward H. Robey. Passive Control of Combustion Dynamics in Stationary Gas Turbines. Journal of Propulsion and Power, 19(5):795–810, September 2003.

[14] Dan Zhao, Zhengli Lu, He Zhao, X. Y. Li, Bing Wang, and Peijin Liu. A review of active control approaches in stabilizing combustion systems in aerospace industry. Progress in Aerospace Sciences, 97:35–60, February 2018.

[15] Michael Stöhr, Isaac Boxx, Campbell D. Carter, and Wolfgang Meier. Experimental study of vortex-flame interaction in a gas turbine model combustor. Combustion and Flame, 159(8):2636–2649, August 2012.

[16] W. Meier, X. R. Duan, and P. Weigand. Reaction zone structures and mixing characteristics of partially premixed swirling CH4/air flames in a gas turbine model combustor. Proceedings of the Combustion Institute, 30(1):835–842, January 2005.

[17] Christoph M. Arndt, Adam M. Steinberg, and Wolfgang Meier. Flame Extinction and Re-Ignition in a Swirl Stabilized Prevaporized Liquid Fuel Flame Close to Lean

Blow-Out. In AIAA Scitech 2020 Forum, Orlando, FL, January 2020. American Institute of Aeronautics and Astronautics.

[18] Christian Kraus, Laurent Selle, and Thierry Poinsot. Coupling heat transfer and large eddy simulation for combustion instability prediction in a swirl burner. Combustion and Flame, 191:239–251, May 2018.

[19] Adam M. Steinberg, Isaac Boxx, Michael Stohr, Wolfgang Meier, and Campbell D. Carter. Effects of Flow Structure Dynamics on Thermoacoustic Instabilities in Swirl-Stabilized Combustion. AIAA Journal, 50(4):952–967, April 2012.

[20] M. Stöhr, C. M. Arndt, and W. Meier. Transient effects of fuel–air mixing in a partially-premixed turbulent swirl flame. Proceedings of the Combustion Institute, 35(3):3327–3335, January 2015.

[21] Patton M. Allison, James F. Driscoll, and Matthias Ihme. Acoustic characterization of a partially-premixed gas turbine model combustor: Syngas and hydrocarbon fuel comparisons. Proceedings of the Combustion Institute, 34(2):3145–3153, January 2013.

[22] Axel Widenhorn, Berthold Noll, and Manfred Aigner. Numerical Characterisation of a Gas Turbine Model Combustor Applying Scale-Adaptive Simulation. In Volume 2: Combustion, Fuels and Emissions, pages 11–23, Orlando, Florida, USA, January 2009. ASMEDC.

[23] Yee Chee See and Matthias Ihme. Large eddy simulation of a partially-premixed gas turbine model combustor. Proceedings of the Combustion Institute, 35(2):1225–1234, January 2015.

[24] Heeseok Koo, Malik Hassanaly, Venkat Raman, Michael E. Mueller, and Klaus Peter Geigle. Large-Eddy Simulation of Soot Formation in a Model Gas Turbine Combustor. Journal of Engineering for Gas Turbines and Power, 139(3):031503, March 2017.

[25] Zhi X. Chen, Ivan Langella, Nedunchezhian Swaminathan, Michael Stöhr, Wolfgang Meier, and Hemanth Kolla. Large Eddy Simulation of a dual swirl gas turbine combustor: Flame/flow structures and stabilisation under thermoacoustically stable and unstable conditions. Combustion and Flame, 203:279–300, May 2019.

[26] Zhi X. Chen and Nedunchezhian Swaminathan. Influence of fuel plenum on thermoacoustic oscillations inside a swirl combustor. Fuel, 275:117868, September 2020.

[27] Ying Huang and Vigor Yang. UNSTEADY FLOW EVOLUTION AND FLAME DYNAMICS IN A LEAN-PREMIXED SWIRL·STABILIZED COMBUSTOR. Begel House Inc., 2003.

[28] Ying Huang and Vigor Yang. Effect of swirl on combustion dynamics in a lean-premixed swirl-stabilized combustor. Proceedings of the Combustion Institute, 30(2):1775–1782, January 2005.

[29] J.C. Broda, S. Seo, R.J. Santoro, G. Shirhattikar, and V. Yang. An experimental study of combustion dynamics of a premixed swirl injector. Symposium (International) on Combustion, 27(2):1849–1856, January 1998.

[30] Cheng Huang, Rohan Gejji, William Anderson, Changjin Yoon, and Venkateswaran Sankaran. Combustion Dynamics in a Single-Element Lean Direct Injection Gas Turbine Combustor. Combustion Science and Technology, 192(12):2371–2398, December 2020.

[31] T. Poinsot. Prediction and control of combustion instabilities in real engines. Proceedings of the Combustion Institute, 36(1):1–28, January 2017.

[32] L Crocco, J Grey, and DT Harrje. On the importance of the sensitive time lag in longitudinal high-frequency rocket combustion instability, 1958. Issue: 12 Pages: 841–843 Publication Title: Jet Propulsion Volume: 28.

[33] T. Schuller, D. Durox, and S. Candel. A unified model for the prediction of laminar flame transfer functions: comparisons between conical and V-flame dynamics. Combustion and Flame, 134(1):21–34, July 2003.

[34] Gowtham Manikanta Reddy Tamanampudi, Swanand Sardeshmukh, William Anderson, and Cheng Huang. Combustion instability modeling using multi-mode flame transfer functions and a nonlinear Euler solver. International Journal of Spray and Combustion Dynamics, 12:175682772095032, January 2020.

[35] Frédéric Boudy, Daniel Durox, Thierry Schuller, Grunde Jomaas, and Sébastien Candel. Describing Function Analysis of Limit Cycles in a Multiple Flame Combustor. Journal of Engineering for Gas Turbines and Power, 133(6):061502, June 2011.

[36] Xingsi Han, Jingxuan Li, and Aimee S. Morgans. Prediction of combustion instability limit cycle oscillations by combining flame describing function simulations with a thermoacoustic network model. Combustion and Flame, 162(10):3632–3647, October 2015.

[37] Matthias Haeringer, Malte Merk, and Wolfgang Polifke. Inclusion of higher harmonics in the flame describing function for predicting limit cycles of self-excited combustion instabilities. Proceedings of the Combustion Institute, 37(4):5255–5262, January 2019.

[38] F. Duchaine, F. Boudy, D. Durox, and T. Poinsot. Sensitivity analysis of transfer functions of laminar flames. Combustion and Flame, 158(12):2384–2394, December 2011.

[39] John L. Lumley and Andrew Poje. Low-dimensional models for flows with density fluctuations. Physics of Fluids, 9(7):2023–2031, July 1997.

[40] W. R. Graham, J. Peraire, and K. Y. Tang. Optimal control of vortex shedding using low-order models. Part II?model-based control. International Journal for Numerical Methods in Engineering, 44(7):973–990, March 1999.

[41] David J. Lucia and Philip S. Beran. Projection methods for reduced order models of compressible flows. Journal of Computational Physics, 188(1):252–280, June 2003.

[42] D. Rempfer. On Low-Dimensional Galerkin Models for Fluid Flow. Theoretical and Computational Fluid Dynamics, 14(2):75–88, June 2000.

[43] M. Bergmann, C. H. Bruneau, and A. Iollo. Enablers for robust POD models. Journal of Computational Physics, 228(2):516–538, February 2009.

[44] Kookjin Lee and Kevin T. Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. Journal of Computational Physics, 404:108973, March 2020.

[45] C. W. Rowley. Model reduction for fluids, using balanced proper orthogonal decomposition. International Journal of Bifurcation and Chaos, 15(03):997–1013, March 2005.

[46] K. Willcox and J. Peraire. Balanced Model Reduction via the Proper Orthogonal Decomposition. AIAA Journal, 40(11):2323–2330, 2002.

[47] Clarence W. Rowley, Tim Colonius, and Richard M. Murray. Model reduction for compressible flows using POD and Galerkin projection. Physica D: Nonlinear Phenomena, 189(1-2):115–129, February 2004.

[48] Eric J. Parish, Christopher R. Wentland, and Karthik Duraisamy. The Adjoint Petrov–Galerkin method for non-linear model reduction. Computer Methods in Applied Mechanics and Engineering, 365:112991, June 2020.

[49] Shady E. Ahmed, Suraj Pawar, Omer San, Adil Rasheed, Traian Iliescu, and Bernd R. Noack. On closures for reduced order models $-$ A spectrum of first-principle to machine-learned avenues. Physics of Fluids, 33(9):091301, September 2021. arXiv:2106.14954 [physics].

[50] Babak Maboudi Afkham, Nicolò Ripamonti, Qian Wang, and Jan S. Hesthaven. Conservative Model Order Reduction for Fluid Flow. In Marta D'Elia, Max Gunzburger, and Gianluigi Rozza, editors, Quantification of Uncertainty: Improving Efficiency and Technology, volume 137, pages 67–99. Springer International Publishing, Cham, 2020. Series Title: Lecture Notes in Computational Science and Engineering.

[51] Kevin Carlberg, Charbel Bou-Mosleh, and Charbel Farhat. Efficient non-linear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations: EFFICIENT NON-LINEAR MODEL REDUCTION. International Journal for Numerical Methods in Engineering, 86(2):155–181, April 2011.

[52] Christopher R. Wentland, Cheng Huang, and Karthikeyan Duraisamy. Investigation of Sampling Strategies for Reduced-Order Models of Rocket Combustors. In AIAA Scitech 2021 Forum, VIRTUAL EVENT, January 2021. American Institute of Aeronautics and Astronautics.

[53] Cheng Huang, Christopher R. Wentland, Karthik Duraisamy, and Charles Merkle. Model reduction for multi-scale transport problems using model-form preserving least-squares projections with variable transformation. Journal of Computational Physics, 448:110742, January 2022.

[54] G Berkooz, P Holmes, and J L Lumley. The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows. Annual Review of Fluid Mechanics, 25(1):539–575, January 1993.

[55] Cheng Huang, Karthik Duraisamy, and Charles Merkle. Challenges in Reduced Order Modeling of Reacting Flows. In 2018 Joint Propulsion Conference, Cincinnati, Ohio, July 2018. American Institute of Aeronautics and Astronautics.

[56] Saifon Chaturantabut and Danny C. Sorensen. Nonlinear Model Reduction via Discrete Empirical Interpolation. SIAM Journal on Scientific Computing, 32(5):2737–2764, January 2010.

[57] Richard Everson and Lawrence Sirovich. Karhunen–Loeve procedure for gappy data. JOSA A, 12(8):1657–1664, 1995. Publisher: Optica Publishing Group.

[58] Benjamin Peherstorfer, Zlatko Drmac, and Serkan Gugercin. Stability of discrete empirical interpolation and gappy proper orthogonal decomposition with random-

ized and deterministic sampling points. SIAM Journal on Scientific Computing, 42(5):A2837–A2864, 2020. Publisher: SIAM.

[59] Ding Li, Sankaran Venkateswaran, Keramat Fakhari, and Charles Merkle. Convergence assessment of general fluid equations on unstructured hybrid grids. In 15th AIAA Computational Fluid Dynamics Conference, Anaheim,CA,U.S.A., June 2001. American Institute of Aeronautics and Astronautics.

[60] Bonnie J. McBride. NASA Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species. National Aeronautics and Space Administration, John H. Glenn Research Center at Lewis Field, 2002. Google-Books-ID: TAEVAQAAIAAJ.

[61] Charles F. Curtiss and Joseph O. Hirschfelder. Transport Properties of Multicomponent Gas Mixtures. The Journal of Chemical Physics, 17(6):550–555, June 1949.

[62] Bruce E. Poling, J. M. Prausnitz, and John P. O'Connell. The properties of gases and liquids. McGraw-Hill, New York, 5th ed edition, 2001.

[63] Philip D. Neufeld, A. R. Janzen, and R. A. Aziz. Empirical Equations to Calculate 16 of the Transport Collision Integrals $^{(l,s)^*}$ for the Lennard-Jones (12–6) Potential. The Journal of Chemical Physics, 57(3):1100–1102, August 1972.

[64] Franck Nicoud, Hubert Baya Toda, Olivier Cabrit, Sanjeeb Bose, and Jungil Lee. Using singular values to build a subgrid-scale model for large eddy simulations. Physics of Fluids, 23(8):085106, August 2011.

[65] G. Erlebacher, M. Y. Hussaini, C. G. Speziale, and T. A. Zang. Toward the large-eddy simulation of compressible turbulent flows. Journal of Fluid Mechanics, 238:155–185, May 1992.

[66] Adam L. Comer, Swanand V. Sardeshmukh, Brent A. Rankin, and Matthew E. Harvazinski. Effects of Turbulent Combustion Closure on Grid Convergence of Bluff Body Stabilized Premixed Flame Simulations. In 2018 AIAA Aerospace Sciences

Meeting, Kissimmee, Florida, January 2018. American Institute of Aeronautics and Astronautics.

[67] Adam L. Comer, Cheng Huang, Karthikeyan Duraisamy, Swanand V. Sardeshmukh, Brent A. Rankin, Matthew E. Harvazinski, and Venke Sankaran. Sensitivity Analysis of Bluff Body Stabilized Premixed Flame Large Eddy Simulations. In AIAA Scitech 2019 Forum, San Diego, California, January 2019. American Institute of Aeronautics and Astronautics.

[68] Charles D. Pierce and Parviz Moin. Progress-variable approach for large-eddy simulation of non-premixed turbulent combustion. Journal of Fluid Mechanics, 504:73–97, April 2004.

[69] K. Seshadri and N. Peters. Asymptotic structure and extinction of methane-air diffusion flames. Combustion and Flame, 73(1):23–44, July 1988.

[70] Norbert Peters. Turbulent combustion. Cambridge university press, 2000.

[71] H Pitsch. FlameMaster: A C++ computer program for 0D combustion and 1D laminar flame calculations, 1998.

[72] M. Frenklach, H. Wang, M. Goldenberg, G. P. Smith, D. M. Golden, C. T. Bowman, R. K. Hanson, W. C. Gardiner, and V. Lissianski. GRI-Mech: An Optimized Detailed Chemical Reaction Mechanism for Methane Combustion. Topical Report, September 1992-August 1995. Technical Report PB96137054, SRI International, Menlo Park, CA.; Pennsylvania State Univ., University Park.; Stanford Univ., CA.; Texas Univ. at Austin.; Gas Research Inst., Chicago, IL., 1995.

[73] S Venkateswaran and Charles Merkle. Dual time-stepping and preconditioning for unsteady computations. In 33rd Aerospace Sciences Meeting and Exhibit, page 78, 1995.

[74] Nicholas Arnold-Medabalimi, Christopher R. Wentland, Cheng Huang, and Karthik Duraisamy. PLATFORM: Parallel Linear Algebra Tool FOr Reduced Modeling. SoftwareX, 21:101313, February 2023.

[75] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. ScaLAPACK Users' Guide. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.

[76] Nicholas Arnold-Medabalimi, Cheng Huang, and Karthik Duraisamy. Large-eddy simulation and challenges for projection-based reduced-order modeling of a gas turbine model combustor. International Journal of Spray and Combustion Dynamics, 14(1-2):153–175, March 2022.

[77] P.L Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. Journal of Computational Physics, 43(2):357–372, October 1981.

[78] Timothy Barth and Dennis Jespersen. The design and application of upwind schemes on unstructured meshes. In 27th Aerospace sciences meeting, page 366, 1989.

[79] I. Boxx, M. Stöhr, C. Carter, and W. Meier. Temporally resolved planar measurements of transient phenomena in a partially pre-mixed swirl flame in a gas turbine model combustor. Combustion and Flame, 157(8):1510–1525, August 2010.

[80] Cheng Huang, William E. Anderson, Matthew E. Harvazinski, and Venkateswaran Sankaran. Analysis of Self-Excited Combustion Instabilities Using Decomposition Techniques. AIAA Journal, 54(9):2791–2807, September 2016.

[81] Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. Journal of Fluid Mechanics, 656:5–28, August 2010.

[82] Peter J. Schmid. Application of the dynamic mode decomposition to experimental data. Experiments in Fluids, 50(4):1123–1130, April 2011.

[83] Nicholas Arnold-Medabalimi, Cheng Huang, and Karthik Duraisamy. Data-Driven Modal Decomposition Techniques for High-Dimensional Flow Fields. In Heinz Pitsch and Antonio Attili, editors, Data Analysis for Direct Numerical Simulations of Turbulent Combustion: From Equation-Based Analysis to Machine Learning, pages 135–155. Springer International Publishing, Cham, 2020.

[84] J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, and Joshua L. Proctor. Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems. Society for Industrial and Applied Mathematics, Philadelphia, PA, November 2016.

[85] WA Light. n-WIDTHS IN APPROXIMATION THEORY (Ergebnisse der Mathematik und ihrer Grenzgebiete 3. Folge, Band 7), 1985.

[86] Benjamin Peherstorfer. Model Reduction for Transport-Dominated Problems via Online Adaptive Bases and Adaptive Sampling. SIAM Journal on Scientific Computing, 42(5):A2803–A2836, January 2020.

[87] Benjamin Peherstorfer and Karen Willcox. Online Adaptive Model Reduction for Nonlinear Systems via Low-Rank Updates. SIAM Journal on Scientific Computing, 37(4):A2123–A2150, January 2015.

[88] Cheng Huang and Karthik Duraisamy. Predictive Reduced Order Modeling of Chaotic Multi-scale Problems Using Adaptively Sampled Projections, February 2023. arXiv:2301.09006 [physics].

[89] Charles K Westbrook and Frederick L Dryer. Chemical kinetic modeling of hydrocarbon combustion. Progress in energy and combustion science, 10(1):1–57, 1984. Publisher: Elsevier.

[90] C. R. Wilke. A Viscosity Equation for Gas Mixtures. The Journal of Chemical Physics, 18(4):517–519, April 1950.

[91] S. Mathur, P.K. Tondon, and S.C. Saxena. Thermal conductivity of binary, ternary and quaternary mixtures of rare gases. Molecular Physics, 12(6):569–579, January 1967.

[92] George Karypis and Vipin Kumar. METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. 1997.

[93] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil M. Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, Jacob Faibussowitsch, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. PETSc Web page, 2022.

[94] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, Jacob Faibussowitsch, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. PETSc/TAO Users Manual. Technical Report ANL-21/39 - Revision 3.18, Argonne National Laboratory, 2022.