Data-Driven Modeling of Compressible Reacting Flow Using Hardware-Oriented Algorithms

by

Shivam Barwey

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

Doctoral Committee:

Professor Venkat Raman, Chair
Associate Professor Jesse Capecelatro
Professor Karthik Duraisamy
Assistant Professor Aaron Towne

Shivam Barwey
sbarwey@umich.edu
ORCID iD: 0000-0002-1717-1805

Dedicated to my parents, Dinesh and Sangeeta Barwey.

ACKNOWLEDGEMENTS

I would first like to thank my family for their invaluable support and encouragement throughout this journey. Thank you for always finding a way to get my mind off of research and de-stress when I most needed it. I always looked forward to coming to Phoenix to visit (in winter, at least!).

This work would not have been possible without the support of the following friends and co-workers. Andy, Chris, and Smriti – thank you for making my transition to graduate school easier and fun. Malik, I am truly grateful for your guidance, especially during my first few years as a doctoral student. I am indebted to you for all of the time you took to help with my research, from high-level ideas to menial code-related issues. Also, thank you for never failing to find the best accommodation options at conferences (I had to say it). Alex, our shared enthusiasm over sports kept me sane that first year when things were especially busy with coursework. Yihao, thank you for answering all of my combustion-related questions and for giving me the best movie recommendations (although I have yet to see Sharknado). Ral, Caleb, Michael, Shivank, and Vansh – thank you for always being ready to get a coffee and (more often than not) a beer. Your company has made the past few years fly by. Ral – I have to thank you specifically for sharing in my suffering related to GPU solver development, and, as much as you refuse to admit, for being my main reference on detonation theory and numerics. Takuma, Negin, and Supraj – I am eternally grateful for your friendship. Negin, our countless walks and conversations in the office were highlights during my time in Ann Arbor. Supraj, there is not much to say at this

point. After the memorable times in undergrad, going through this roller-coaster of a Ph.D experience with you has been an honor. I will never forget the endless nights working on papers, proposals, presentations, and codes – especially during the disaster that was 2020!

I would like to thank the members of my dissertation committee – Professors Karthik Duraisamy, Aaron Towne, and Jesse Capecelatro – for taking the time to provide valuable technical feedback and advice related to the topics covered in this dissertation.

Lastly, I must thank my advisor, Professor Venkat Raman. None of this would have been possible without your motivation, encouragement, guidance, and patience. Thank you for giving me the freedom to explore a wide variety of research problems, and for always being open to discuss any idea, no matter how ridiculous or far-fetched (looking back, this was often the case). You have taught me a great deal within and beyond research – too much to list here. All I can say is that the dedication you show to your work and to your students has inspired me immensely. I am truly lucky to have you as an advisor and a friend.

TABLE OF CONTENTS

CHAPTER

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

LIST OF ABBREVIATIONS

AI Artificial intelligence

AIM Approximate inertial manifold

AMR Adaptive mesh refinement

ANN Artificial neural network

CFD Computational fluid dynamics

CFL Courant-Friedrichs-Lewy

CJ Chapman-Jouguet

CNS Compressible Navier-Stokes

CPU Central processing unit

CROM Cluster-based reduced order modeling

CSP Computational singular perturbation

DMD Dynamic mode decomposition

DNS Direct numerical simulation

EMD Earth mover's distance

FLOPs Floating point operations

FPV Flamelet/progress variable

GPU Graphics processing unit

HPC High-performance computers / high-performance computing

ISAT In-situ adaptive tabulation

JS Jacobian-scaled

LES Large-eddy simulation

LMDE Linearized model detonation engine

ML Machine learning

MPI Message passing interface

MSE Mean-squared error

ODE Ordinary differential equation

PDF Probability density function

PDE Partial differential equation

PINN Physics-informed neural network

POD Proper orthogonal decomposition

PRISM Piecewise reusable implementation of solution mapping

PVC Precessing vortex core

QOI Quantity of interest

RDE Rotating detonation engine

RHS Right-hand side

ROM Reduced-order model

SIMT Single-instruction multiple-thread

TVD Total variation diminishing

WCSS Within-cluster sum of squares

ZND Zeldovich-von Neumann-Döring

ABSTRACT

High-fidelity numerical simulations of combustion processes in next-generation hypersonic propulsion devices (including, but not limited to, rotating detonation engines and scramjets) play a crucial role in enabling robust design strategies for their real-world deployment. These simulations, however, require full-geometry numerical solutions of the compressible reacting Navier-Stokes equations. Spatiotemporal resolution requirements stemming from multi-scale interactions between turbulence, shockwaves, and chemical reactions contained in these governing equations induce computationally prohibitive bottlenecks that render the required long-time resolved simulations of these propulsion devices infeasible. A particularly elusive bottleneck comes from the treatment of detailed chemical kinetics required to accurately describe the time evolution of species concentrations and flow-chemistry interactions within the combustors. The computational hurdles emerge here from the arithmetic intensity of chemical source term evaluations and immense disparities in chemical timescales for practical fuels.

The goal of this work is to provide a physics-guided data-driven modeling strategy for accelerating high-fidelity compressible reacting flow solvers via elimination of the chemistry bottleneck. Since unsteady features of interest in compressible reacting flow (e.g. detonations) are sustained by chemical reactions, the principle assumption is that local regions in the thermochemical state space can be used to classify spatially coherent regions of dynamical similarity within the reacting flowfield in physical space. Based on this assumption, the modeling approach finds these local regions using an unsupervised clustering algorithm and deploys targeted models for accelerated chemical source term evaluation within each region. The novelty comes from (a) ensuring that flowfield classifications enabled by the clustering procedure are consistent with physical expectations in complex compressible reacting flow (e.g. the clusters identify meaningful regions within detonation wave structure in rotating detonation engines), and (b) embedding physical knowledge directly into the clustering objective function. Emphasis is placed on ensuring the modeling framework can be extended to in-situ (or online) integration with flow solvers, such that the method is not tied down to single geometric configurations. Additional steps are taken to ensure that the algorithms used in the modeling approach are compatible with modern high-performance computing trends dominated by GPU-centric node architectures.

CHAPTER I

Simulations of Hypersonic Propulsion Devices

1.1 Introduction

The unabated evolution of computing power, supplemented by the accessibility of easy-to-use software development frameworks in modern times, has cemented the role of numerical simulations as a reliable method of scientific inquiry in a myriad (if not all) of natural sciences. Continued progression of compute capability, and the subsequent evolution of simulation fidelity, is evidenced in part by the predictions of Moore's law, shown in Fig. 1.1, which comes from the empirical observation that the number of transistors in a given microchip doubles approximately every two years [210][1]. If one is to isolate a single effect of the ubiquity of simulations in the 21st century, it is that field experts now have a means to generate an exorbitant amount of synthetic data to supplement real-world observations obtained from traditional experiments. In other words, the big-data phenomenon often used to characterize modern times is ultimately made "bigger" by these simulations by means of leveraging the centuries of underlying physical knowledge of the respective systems under investigation.

In all fields, research and development funding and manpower targeted at im-

[1] Despite some claims that the progression of Moore's law is set to decay in the coming decades, continued breakthroughs in hardware are expected to maintain the steadfast rise of compute capability [309].

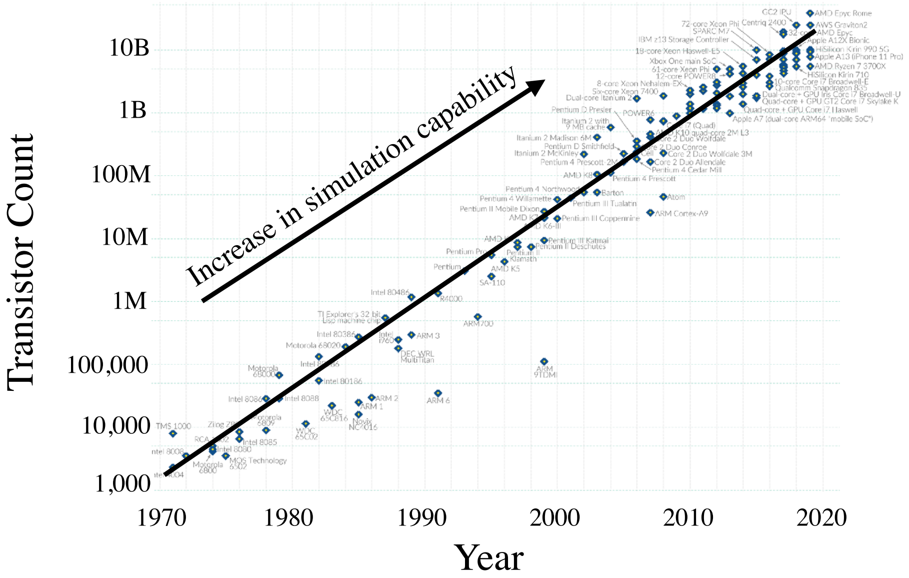


Figure 1.1: Visualization of Moore's Law in logarithm scale, where scatter points reflect commerically-available hardware and black line reflects linear fit (image licensed under CC-BY from https://ourworldindata.org).

proving the potency of these numerical simulation codebases depend entirely on the physical complexity of the set of objects or phenomena in study. As a consequence, it is to the interest of all members of the given field that these simulations produce results that are not only physically consistent (i.e. the simulation-generated data can be trusted to some quantifiable level of error), but are also available at a fast turnaround time, which can be on the order of minutes, hours, or days depending on levels of complexity, industry requirements, and available computing/user resources.

As computing platforms and state-of-the-art hardware become more power efficient, the scope of numerical simulations in tandem becomes increasingly ambitious in terms of physical complexity. A primary example, which is the focus of this work, is the rapid research progress made within the field of computational fluid dynamics (CFD); numerical simulation tools that utilize CFD, termed colloquially as flow solvers, are readily used these days to model and characterize so-called multi-physics phenomena. As the term implies, applications and industrial devices that exhibit multi-physics behavior are characterized by an amalgamation of physical phenomena that operate at a wide range of length and time scales – flow solvers are deemed "high-fidelity" are capable of resolving the effect of these multi-scale interactions within the simulation process. In other words, within fluid dynamics applications, multi-physics problems imply that the application or device of interest is described by the interactions of more than one form of underlying physical source. As emphasized further below, these sources can include turbulence, chemical reactions, electromagnetism, gravity, and shockwaves, among other effects.

Because of these interactions, the research challenge comes from (a) understanding physically how devices exhibiting multi-physics behavior operate and (b) creating prognostic simulation frameworks using the extracted physical knowledge. Naturally, advances in the predictive capability for multi-physics flow, as well as its physical understanding, is enhanced and spurred by the back-and-forth feedback between these two challenges. Engineering advances resulting from addressing these challenges hinge on the fact that the wide ranges in length and timescales emanating from the interactions of the varying multi-physical sources need to be accounted for in simulations, and well-understood physically.

To this end, multi-physics flow simulations have emerged as crucial tools used in a plethora of application fronts particularly relevant to the defense, energy, and national security industries – some examples of applications that gain significant benefit from the utilization of high-fidelity multi-physics simulation output include, but are not limited to, description of pollutant dispersion in urban environments (Fig. 1.2(a)), thermonuclear reactor design (Fig. 1.2(b)), and development of next-generation propulsion devices (Fig. 1.2(c)). As implied by these applications, the output of these simulations is in general most valuable when either little physical knowledge about a device can be extracted using existing experimental technology, or the potential risks/expenses in exploring a design space for the device in the real-world are too high.

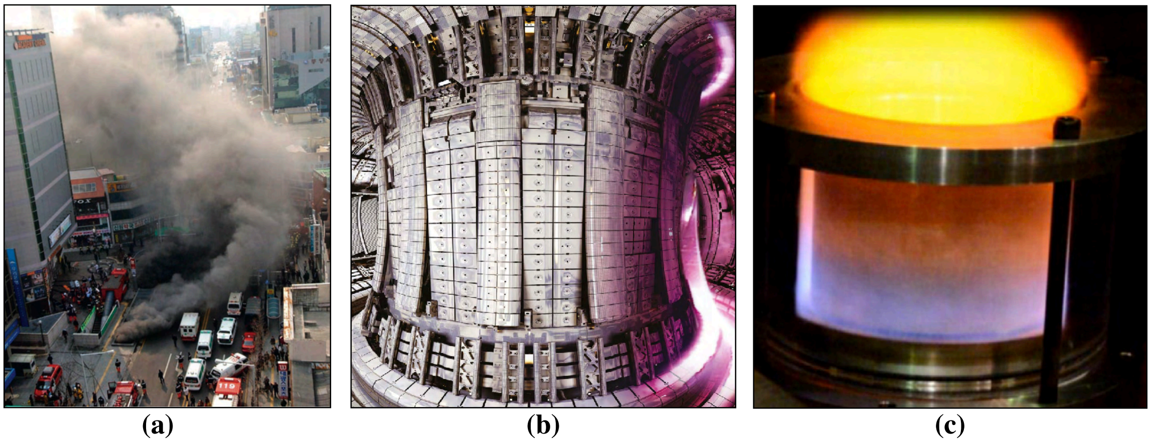


Figure 1.2: Examples of modern applications that leverage multi-physics simulation. **(a)** Prediction of contaminant dispersion in urban environments (from Ref. [233]). **(b)** Determining safety bounds for Tokamak reactors, which are energy generation devices based on nuclear fusion (from Ref. [104]). **(c)** Designing rotating detonation engines for hypersonic propulsion (from Ref [282]).

1.2 Next-Generation Hypersonic Propulsion Devices

Of all these applications, the scope for this dissertation is narrowed to the design and development of next-generation propulsion devices governed by compressible reacting flow, which indeed admits the aforementioned complex multi-physics behavior via interactions between shocks, chemical reactions, and turbulence (or turbulent combustion). For context, two types of devices immediately relevant to the field of hypersonic propulsion, namely rotating detonation engines (RDEs) and scramjets, are described briefly in the paragraphs below, as the primary goal of compressible reacting flow simulations is to properly predict and characterize the behavior of such devices in an accessible way to guide their designs.

The goal of the rotating detonation engine, a schematic of which is shown in Fig. 1.3, is to utilize detonations instead of the more commonly used combustion mode of deflagration to produce net pressure gains and therefore higher thrust outputs [181, 327]. Detonation waves are, in broad terms, self-sustained shock fronts that are tightly coupled with reaction zones [86]. One can show, using fundamental physical principles of compressible reacting flow, that the speed at which the detonation wave travels (termed the Chapman-Jouguet (CJ) speed) is a function of both the ambient conditions of the gas and the composition (or description) of the fuel and oxidizer mixture [106]. In the typical RDE, one or more detonation waves traveling at this CJ speed (which can readily exceed thousands of meters per second) propagate azimuthally in an annular chamber, processing a mixture of fuel and air injected axially (or radially, depending on the configuration) from the bottom of the device. Due to the unsteady nature of the self-sustained detonations, as well as complexities in fuel injection schemes, the combustion processes within an RDE are known to be highly chaotic, often departing from canonical detonation behavior [304]. Detonations observed in RDEs are non-ideal [278], since the reaction zones within the detonation wave structure are significantly broader than the theoretical expectations derived from simpler steady-state solutions attributed to the idealized Zeldovich-von Neumann-Döring (ZND) structure [225]. These complexities and non-idealities are exacerbated by the interaction of many flow features and combustion regimes, such as (a) combustion zones ahead of the wave that perturb the detonation dynamics (termed parasitic deflagration) [52, 53], (b) the oscillation of high-pressure regions (triple points) at the wavefront due to collisions between the principle shockwave and an erratic distribution of weaker transverse shockwaves [114, 173, 195], and (c) slow, decayed heat release that extends far behind the wavefront, homogenizing the mixture of chemical composition at fast timescales [298, 355]. Physical analysis of RDEs in various formats, from both experimental and high-fidelity numerical perspectives [295], has focused on evaluating the effects of mass flow rates in the injection scheme [297], understanding impacts of different industry-relevant fuel compositions [274, 281], and studies of non-idealities in the detonation wave structure due to geometric effects such as annulus curvature [14, 171, 278] and perturbations induced by turbulence-chemistry interactions [97, 271, 272, 298].

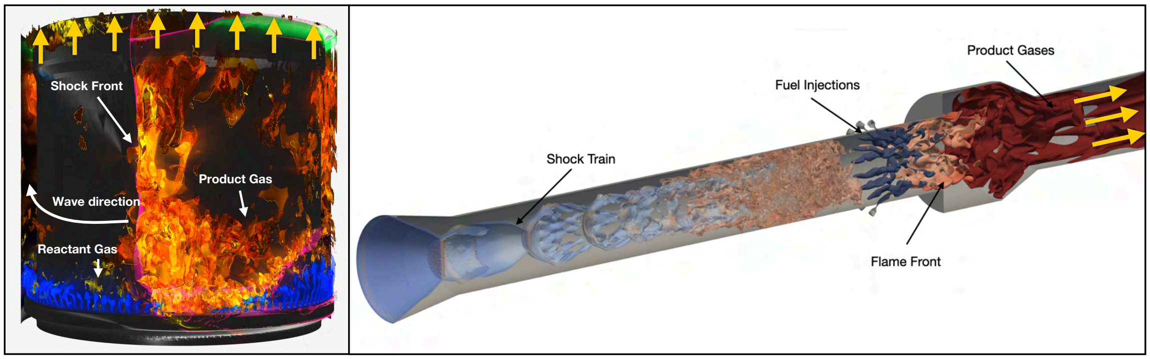


Figure 1.3: **(Left)** Instantaneous flowfield within a rotating detonation engine [274]. **(Right)** Instantaneous flowfield within a scramjet [30]. Thrust vectors are given by the yellow arrows, and major flow features within each device are labeled in the respective images. Both images were generated using rendered snapshot data from UM-ReactingFlow [30], a high-fidelity compressible reacting flow solver for full-geometry combustors developed at the University of Michigan.

On the other hand, scramjets (supersonic combustion ramjets), a schematic of which is also shown in Fig. 1.3, are air-breathing propulsion systems that utilize a series of shock-waves to decelerate the flow (the compression stage) until it reaches the combustion chamber, in which fuel is injected and subsequently mixed to induce chemical reactions and produce thrust downstream [59, 67]. In contrast to ramjets, which observe flow deceleration to subsonic levels during the shock train-induced compression stage, combustion in scramjets occurs in the supersonic regime [98]. As such, similar to RDEs, the unsteady combustion processes are highly nonlinear, advection-dominated, and amenable to high levels of instability. However, in contrast to RDEs, the mixing process in scramjets play a major role: as fuel is supplied to the combustion chamber in an injection phase, the chaotic mixing processes and recirculation zones at supersonic conditions gives rise to complex auto-ignition and flame behavior that is notoriously difficult to predict [35, 305]. Physical analysis of scramjets has focused on the effect of thermochemical models on autoignition propensity [89, 136], relations between boundary layer development and the shock train [88, 95], and general thermoacoustic instabilities emanating from the chaotic combustion processes induced by acoustic wave reflections within the device [189, 247].

1.3 Role of High-Fidelity Simulations

Because of the exciting theoretical potential of hypersonic propulsion and energy-generation devices like RDEs and scramjets, frameworks for optimizing their design for real-world testing are essential. As such, design exploration for all devices integrating these propulsion concepts is a highly active area of research within academia, government-sponsored labs, and the defense industry. Fast iterations on various design considerations for these devices – e.g. fuel injection angles for RDEs and cavity geometries for scramjets – are required in order to bring these devices into the mainstream.

As suggested by the descriptions in Sec. 1.2, one of the primary research challenges in the design process for these propulsion devices lies in the physical characterization of, and modeling strategies for, combustion phenomena described by the evolution of compressible reacting flow [276]. In other words, the design of future energy generation and propulsion devices requires characterizing their macroscopic behavior – this can be specific impulse, thrust, pollutant generation, or some other performance-related quantity of interest (QOI) – for industry-relevant operating conditions and various geometric configurations. Through the development of robust flow solvers, high-fidelity numerical simulations are used in the iterative design process alongside real-world experiments of model configurations. On one hand, experiments, which are typically constrained to single model geometries, are used to validate the numerical implementations of flow solvers and supply informative boundary conditions. On the other hand, the then validated high-fidelity simulation codebase can be used to extrapolate to different geometric configurations that ultimately inform the end-user or customer, by means of detailed flowfield analysis, of real-world viability. Although commissioning high-fidelity simulations can be expensive, the crux of this procedure is to minimize the development of as many experimental configurations as possible, which is generally more cost and labor-intensive than running simulations. Figure 1.4

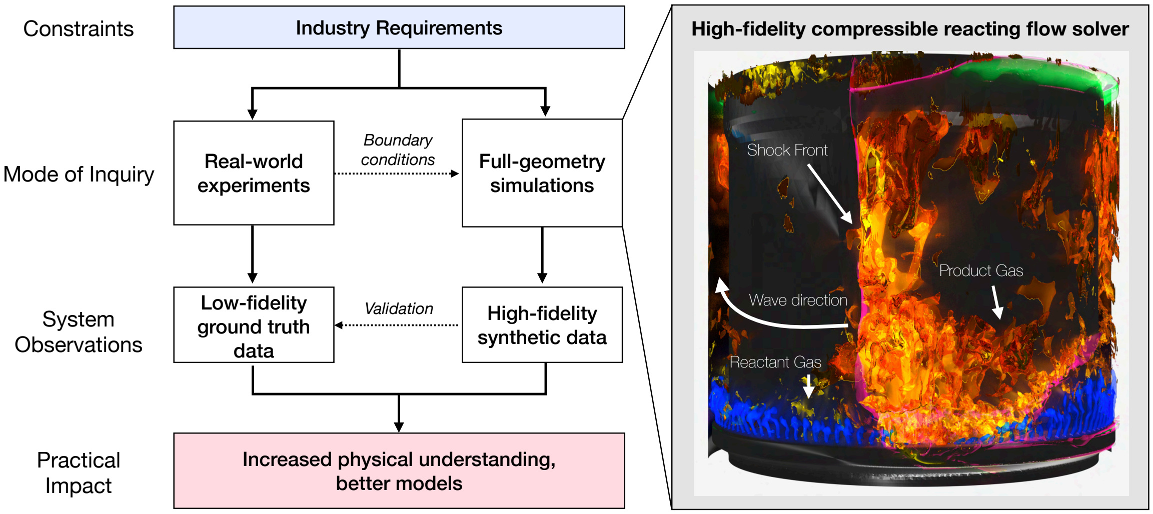


Figure 1.4: The scope of high-fidelity, full-geometry simulations in the application context of driving better designs for propulsion devices like RDEs.

illustrates this scope of high-fidelity numerical simulations and associated feedback with experiments, where the idea is to probe as many industry-relevant operating conditions as possible. In the end, both simulations and experiments produce valuable data that can be used by field experts to not only increase physical understanding of the devices or physical features in question, but also to produce robust modeling frameworks that accelerate the exploration of the combustor design space.

It should be emphasized that detailed exploration of design envelopes for propulsion devices like RDEs and scramjets from a purely experimental or real-world standpoint without simulations is inherently not only high-cost due to geometric complexities, but also high-risk due to the nonlinearities associated with multi-physics interactions stemming from the governing equations of compressible reacting flow (the Navier-Stokes equations). In other words, accurate trends in macroscopic QOIs required to characterize practical design envelopes for these applications are only recovered by capturing the small-scale, chaotic interactions between diffusion-dominated processes (turbulence), chemical reactions (species production rates), and advection-dominated processes (acoustic waves and shock waves of varying strength). When

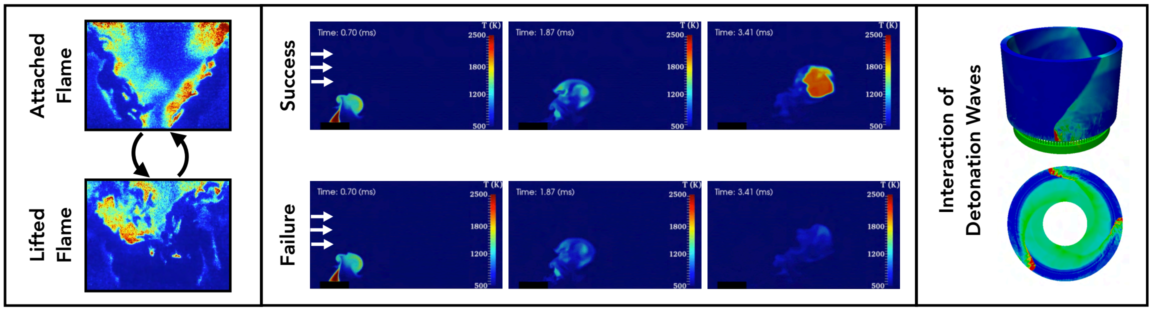


Figure 1.5: Examples of extreme events in combustion. **(Left)** Spontaneous flame shape transition in lean premixed operating conditions (from Ref. [10]), **(Middle)** High-altitude relight/ignition failure due to indistinguishable variations in turbulent inflow (from Ref. [123]). **(Right)** Wave-splitting phenomena in RDEs, where one initial wave evolves into two or more for same operating condition (from Ref. [297]).

considering target operating condition requirements such as lean premixed combustion at high-altitudes, supersonic combustion, and sustained detonation, these non-linearities can give rise to seemingly spontaneous changes in the system state that drastically alter device operation behavior. Examples of such state changes present in combustion processes relevant to propulsion applications are provided in Fig. 1.5; these are known formally as extreme events [119, 122], and include (a) the spawning of multiple detonation waves from a single detonation wave in RDEs, (b) spontaneous flame shape transitions due to stringent equivalence ratio and fuel requirements in power generation devices, and (c) high-altitude relight failures in scramjets. Due to the fact that undetected and unmitigated extreme events can very well result in irreparable structural damage, costly changes to maintenance/repair/overhaul (MRO) schedules, and potentially loss of human life, a fail-safe and low-risk design iteration loop must be facilitated by high-fidelity numerical simulations that accurately describe the multi-physics behavior of the devices in question.

As illustrated in Fig. 1.4, in an example scenario, a user may commission full-geometry simulations of a model combustor in a first step. Then, in a second step, the validated simulation tools can be used to extrapolate to new designs as needed to accumulate and saturate relevant QOIs in the design space. Overall, within the context of such design frameworks, the above discussion emphasizes the crucial role high-fidelity simulations play – the downside, however, is that ensuring all levels of complexity in devices like RDEs and scramjets are resolved in numerical implementations is generally a computationally prohibitive task. The next section gives background on some of these inevitable computational challenges.

1.4 Computational Challenges for Reacting Flow Solvers

As discussed in Sec. 1.3, one of the drivers of robust design strategies for improving and deploying hypersonic propulsion devices like RDEs and scramjets is the commissioning of high-fidelity, full-geometry compressible reacting flow simulations at industry-relevant operating conditions. In this context, high-fidelity treatment amounts to resolved numerical predictions of flowfields inside of the combustion chambers of these devices of interest, the dynamics of which are governed by the compressible reacting Navier-Stokes equations. For these applications, ”resolved” refers to the fact that all all length and timescales are properly accounted for in the numerical solutions. In other words, the governing equations must be directly solved by means of numerical discretization schemes without the usage of models for residual forcing terms. This is known as direct numerical simulation (DNS) [209], which is a highly computationally intensive task in situations where there is a large disparity in length and time-scales.

In full-geometry DNS of compressible reacting flow, computational challenges come from two primary sources: (1) the stringent spatial resolution requirements associated with turbulent lengthscales, detonations, and shock-fronts, and (2) the prohibitive temporal resolution requirements that come from very fast chemical reactions required to correctly predict species production and consumption rates. An additional challenge comes from the fact that the complex geometry itself must be represented numerically in a meshing phase – the numerical treatment of complex geometries has significant impact on how one designs discretization schemes to treat the above two challenges related to spatiotemporal resolution, and as a result, contributes significantly to general computational challenges in conducting accurate simulations of industrial burners. These three aspects – lengthscale requirements, timescale requirements, and complex geometry treatment – will be described in more detail in the paragraphs below. The discussion will be tailored to the application scope of RDE and scramjet simulations, but the discussion holds in general for any application that is described by compressible reacting flow. Before proceeding, it should be noted that alongside the numerical challenges induced by the underlying physics that define the governing equations of interest, the programming challenges required to develop a reliable production-level solver or code to carry out such simulations is in itself a significant effort that requires several months, if not years, of software development effort. Ensuring algorithms are properly verified and validated is a monumental task in itself – detailed discussions on these types of topics are out of scope here (see Refs. [15, 283, 294] for more information on these subjects).

1.4.1 Spatial Resolution Constraints

Many of the relevant spatial flow features in devices like RDEs and scramjets, which are driven by advection-based compressible flow phenomena, occur over very small lengthscales which in turn induces high numerical resolution penalties [163, 278]. Accurate numerical representation of shockwaves, for example, presents a difficult computational problem for high-fidelity simulation of any supersonic or hypersonic propulsion device. Because the shockwave is inherently discontinuous, the challenge comes from ensuring that the discretization of advection terms, as well as treatment of numerical fluxes, must be not only physically consistent (e.g. kinetic energy-preserving), but also numerically stable such that the flowfields are not polluted by numerical errors [335]. This requirement traditionally amounts to utilizing low-order numerical schemes that are highly dissipative near shock and contact discontinuities in order to ensure spurious oscillations characteristic of higher-order and/or dispersive numerical schemes are not generated [64]. Ultimately, stability requirements in production-level compressible flow solvers forces the user to (a) use globally low-order numerical schemes (e.g. TVD schemes) and refine the mesh to exceedingly high levels such that numerical diffusion is diminished in all regions in space [23, 234], (b) utilize complex hybrid schemes that are dissipative only near shock or contact discontinuities and preserve kinetic energy [54, 257], but require ad-hoc and non-intuitive parameter tuning steps that may introduce other physically inconsistent flow effects, or (c) set aside the low-order schemes entirely and use expensive, locally adaptive high-order schemes with explicit filtering steps to directly eliminate non-physical dispersive errors that tend to perturb boundary conditions to unacceptable levels [55, 249, 333]. Regardless of which path is taken, due to the prescence of shockwaves alone, all three of the above scenarios lead to significant added computational expenses for high-fidelity compressible reacting flow solvers.

A key example for RDE-based applications is the sufficient resolution of detonation waves, which is interpreted in general terms as shockwave sustained by chemical reactions, and is the driving flow feature for predicting macroscopic device operation trends [311]. In other words, for the solver to capture correct theoretical detonation wave speeds, one must properly discretize the spatial domain within the detonation wave to acceptable levels. This discretization requirement is problematic not only because of high levels of unsteadiness and chaoticity present within the wave itself in unsteady settings, but also because the relevant detonation lengthscales are also a function of the parametrization of chemical heat release (e.g. the chemical kinetic model) [270, 313].

Figure 1.6 shows the output of an idealized hydrogen-air detonation wave in one spatial dimension obtained from a steady-state ZND solution of the compressible

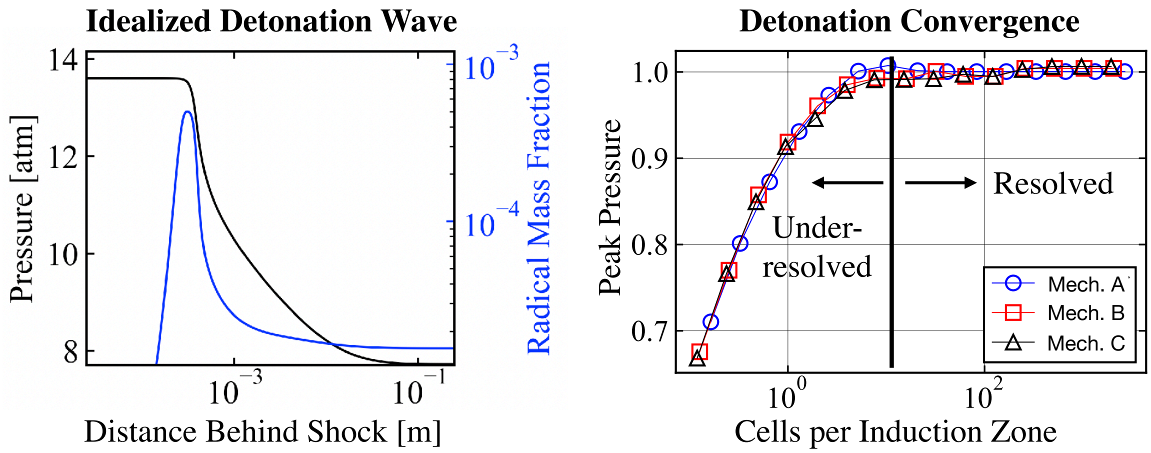


Figure 1.6: **(Left)** Idealized detonation wave structure obtained from steady state 1d solutions to the compressible Navier-Stokes equations using the hydrogen-air mechanism of Burke et al. [45]. Plot is in shock reference frame, such that x=0 corresponds to the shock front – black is pressure and blue is HO_2 mass fraction. **(Right)** Peak detonation pressure normalized by analytic value versus grid resolution obtained from unsteady detonation simulations for three different mechanisms (from Ref. [17]).

Navier-Stokes equations[2]. This figure illuminates the three flow features that must be sufficiently numerically represented for RDE simulation: (a) the shockwave, which constitutes a jump condition from the ambient state to the compressed post-shock state, (b) the induction zone, defined as the distance behind the shock-front at which the flow features remain fixed at the post-shock conditions (region of constant pressure in Fig. 1.6), and (c) the reaction zone, defined as the lengthscale required to consume the fuel at a rate defined by certain kinetics-induced timescales (indicated by the radical species generation in Fig. 1.6).

In practical applications, unsteady detonations deviate from the steady state solutions [278] – despite this, the idealized wave structure shown in Fig. 1.6 serves as a guideline for convergence studies and mesh generation, as it provides the baseline resolution requirements for detonation simulation. In the end, to guarantee that the correct unsteady detonation dynamics are represented within RDEs (e.g. the interaction between detonation and deflagration regimes, movement of pressure waves within

[2] See Sec. 2.2 for a description of the governing equatinos. The reader is referred to Chapter 5 in the textbook by Glassman and Yetter [106] for details on ZND and fundamental detonation theory.

the induction zone, and correct radical generation in the reaction zone), the computational grid must ensure that some cutoff number of cells – typically on the order of 10 to 100, depending on the combustion properties of the fuel-oxidizer mixtures – are contained within the detonation wave structure at all times [17, 191, 313]. The associated lengthscales for industry-relevant fuels renders this requirement very expensive, particularly in cases where complex geometric configurations are required and when long simulation times are desired. For example, failure to properly resolve the wave structure, as shown in Fig. 1.6, results in incorrect prediction of detonation wave properties, which eventually cascades into inaccurate and misleading representations of important design parameters for these devices (e.g. thrust, specific impulse, peak pressures loads, etc.).

Although the key flow features in propulsion devices like RDEs and scramjets are characeterized by advection-dominated reacting flow, diffusion-dominated features also play key roles and introduce additional lengthscale requirements on these simulations related to turbulence and mixing. In RDEs, for example, the stability of the detonation wave, as well as the propensity to generate additional detonation waves from a single wave, is highly influenced by the fuel-air mixing process in the injector array, which is driven by the high shear generated from the injection scheme (this is why injection angles are key design parameters in hypersonic propulsion applications in general). Hence, capturing the turbulent mixing mechanisms and their interplay within the unsteady detonation process is crucial, but expensive due to traditionally prohibitive turbulent lengthscale resolution requirements [274, 278, 298]. Similar requirements hold for scramjet simulations, for which high resolution of boundary layers are required to properly capture shock interactions and pressure loads at the boundaries [88, 159]. Additionally, accurate numerical representation of the impact of turbulence on combustion processes within these devices is of paramount importance overall – the impact of reacting scalars on the turbulent energy spectra in mixing-dominated regions must be accounted for by means of high levels of numerical resolution. Since these mixing dominated features may not be local in space, addressing these requirements is highly nontrivial and often results in very large meshes that slow down computations. For context, within RDEs, turbulent combustion plays a key role in detonation wave structure dynamics via fuel stratification effects – the detonation wave itself induces an energy spectrum in the reacting scalars parametrized by an integral lengthscale, such that the stability of subsequent passes of the detonation wave is a function of this lengthscale [271, 272, 336]. On the other hand, in scramjets, turbulent combustion effects manifest in the fuel injection, mixing, and autoignition phenomena after the shock-induced compression of the flowfield [159].

1.4.2 Temporal Resolution Constraints

Ensuring robust direct numerical simulations of complex propulsion devices goes beyond addressing the spatial resolution constraints discussed above in isolation. Temporal constraints also prohibit high-fidelity simulation capability. In the same spirit of ensuring accuracy and stability by means of stable spatial discretization methods and high grid resolution, the same ideas must also be held in temporal discretization [110].

In many cases, physical constraints for the simulation time step in advection-dominated systems are provided by the Courant-Friedrichs-Lewy (CFL) condition [65], which is a mathematical rule that provides the highest timestep that can be used within the flow solver (here assumed to be utilizing explicit time-marching methods) while preserving physical bounds of information propagation [335]. This required timestep scales linearly with the local spatial grid size and inversely with wavespeed estimates – in other words, higher grid resolutions produce smaller required timesteps to ensure stability, and higher wavespeeds compound this effect. As such, since devices like scramjets and RDEs observe both high spatial resolution requirements as well as large local wavespeeds (detonation velocities can routinely exceed 1500 m/s, for example [278]), the advection-based limitations on temporal resolution naturally creates significant computational hurdles.

For devices described by compressible reacting flow, the impact of chemical reactions on the flowfield must also be properly accounted for in the governing equations [296]. In the end, the most accurate representations of chemistry in the governing equations provided by so-called detailed chemical kinetic descriptions [90, 183]. In detailed kinetics, the multi-physics behavior stemming from chemical reactions induces a highly complex Arrhenius-based expression for the species source terms, the nonlinearity of which produces an extremely stiff set of equations for the time evolution of the reacting chemical species [261]. These detailed kinetic descriptions, referred to colloquially in the numerical combustion community as "detailed chemistry", are experimentally validated models for species production rates that consist of a collection of physically constrained elementary chemical reactions. Although these are technically models, these are referred to as the ground-truth in numerical applications due to the fact that they are the most accurate baseline for representing the highly nonlinear effects of chemical reactions on the flowfield numerically [183, 258]. Depending on the target fuel and oxidizer, these detailed chemistry descriptions require transporting and storing on the order of tens, hundreds, or thousands of additional transport equations per computational grid point (one for each species), which amounts to increasing the simulation degrees of freedom by at least one order of magnitude. Alongside the added storage requirements and addition of more transport equations, these Arrhenius-derived nonlinearities induce a wide range of timescales that must be resolved to accurately represent the characteristic turbulence–chemistry interactions required to properly predict macroscopic trends driven by unsteady processes in devices like RDEs and scramjets. This disparity in chemical timescales, known as chemical stiffness [218], is a universal bottleneck in high-fidelity reacting

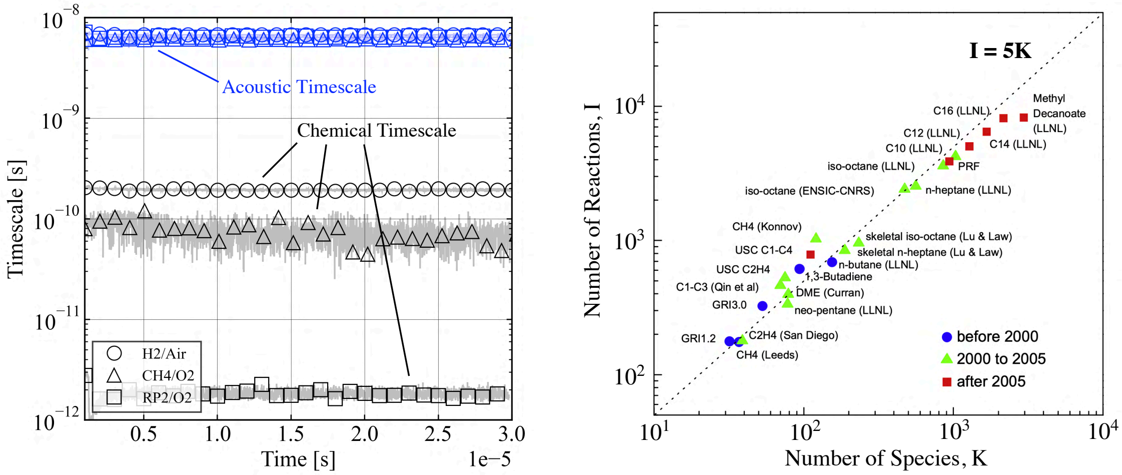


Figure 1.7: **(Left)** Comparison of acoustic versus chemical timescales obtained at detonation fronts in high-fidelity RDE simulations using three different chemical mechanisms (adapted from Ref. [30]). **(Right)** Plot of number of reactions versus number of species for detailed chemical mechanisms used in high-fidelity reacting flow simulations (adapted from Ref. [183]).

flow simulations.

To emphasize the impact of chemistry-related timescale restrictions, a comparison between acoustic timescales and chemical timescales extracted from numerical simulations of detonation waves for three different detailed kinetic mechanisms of varying degrees of complexity are shown in Fig. 1.7. Acoustic timescales are obtained using standard wavespeed estimates, and chemical timescales at the wavefront are extracted from the chemical Jacobian eigenvalues [190]. The trends in Fig. 1.7 emphasize the sensitivity of chemical stiffness to the fuel/oxidizer combination and illuminate the chemistry-derived bottlenecks that arise in the simulation of propulsion devices like RDEs. In the case of industry-relevant fuels like RP2 [360], the chemical timescales are several orders magnitude smaller than the CFL-derived acoustic timescales.

An important point is that chemistry treatment required in these flow solvers does not only add stiffness, but also adds the evaluation of an additional source term on the right-hand-side (the chemical source term). Numerical evaluation of the chemical source term itself is a computationally intensive task due to the high arithmetic intensity of the Arrhenius formulation (see Sec. 2.2); a major issue is that this complexity not only scales inversely with the limiting timestep (the number of RHS evaluations increases as the timestep decreases), but also linearly with the number of species retained in the kinetic mechanism [16, 183]. As a result, increasing the accuracy of the representation of chemistry in the flow solver, which effectively amounts to increasing the number of species, translates ultimately to a significant increase in computational cost required to evaluate the chemical source terms. This phenomenon, illustrated in Fig. 1.7, comes from the physical constraints of elementary reactions which are the building blocks for these detailed mechanisms. For a given chemical mechanism, Fig. 1.7 shows how increasing the number species results in a 5X increase in the number of reactions, which amounts to a superlinear scaling of source term evaluation times.

1.4.3 Treating Complex Geometries

The above computational challenges are exacerbated when treating complex geometries that produce complex, irregular meshes[3]. In general, there are two options to integrate complex geometries into simulation workflows: (1) utilize unstructured grids [219], or (2) utilize an immersed or embedded boundary method, which represents implicitly the effect of a complex geometry on a structured grid by means of storing additional level set scalar fields [200, 250]. An example of the output of unstructured meshing required to discretize the interior domain of an injector in a rotating detonation engine simulation is shown in Fig. 1.8. Complexities induced by these methods on both boundary condition treatment and data structure arrangement eventually add to computational costs, especially for numerical flux evaluation. For example, in unstructured meshes, the connectivity matrix that assigns cell centroids to cell

[3] As discussed in the CFD vision 2030 study, it should be noted that the meshing procedure is in itself a very time-consuming part of the workflow, and can take days to weeks depending on the resolution requirements and available computer storage/memory [317].

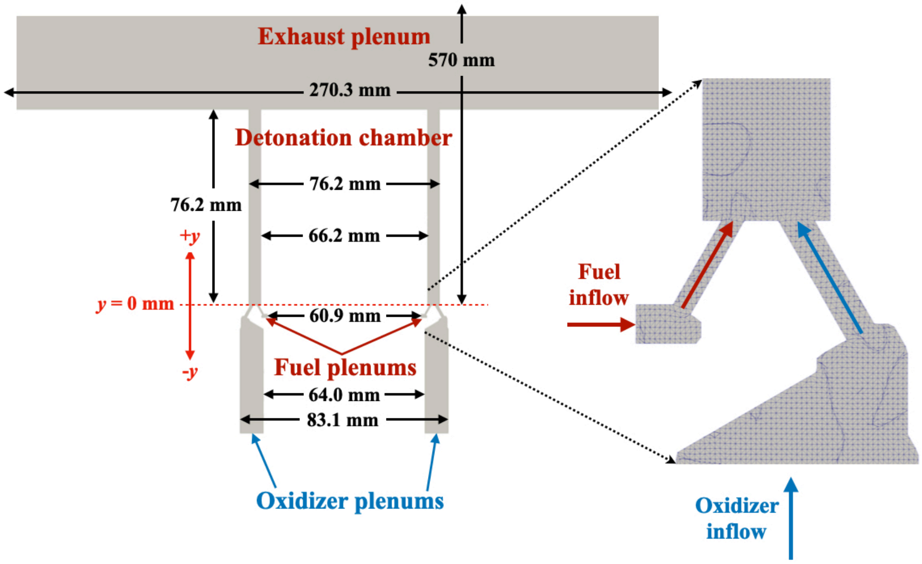


Figure 1.8: Mesh used for a single fuel injector geometry within a rotating detonation engine simulation (from Ref. [30]).

faces is very large and irregularly organized in memory [30] – as such, computing surface integrals by means of cell-to-face interpolation and the Gauss divergence theorem, which is a backbone routine for finite volume methods [335], inevitably adds to computational cost when treating irregular geometries required to explore advanced propulsion concepts.

1.5 Overview and Scope of Dissertation

The discussion in Sec. 1.3 and 1.4 brings forward two conflating requirements: the first is that simulations of propulsion devices must utilize complex geometry and high-fidelity numerical solutions of compressible reacting flow that present computationally prohibitive bottlenecks, and the second is that a large number of long-time simulations must be commissioned for robust exploration of design spaces. As such, this has led to immense research in exploring pathways for *simulation acceleration* – that is, strategies for addressing these computational challenges by eliminating the critical bottlenecks/challenges outlined in Sec. 1.4.

As illustrated in Fig. 1.9, there are two such pathways by which one can accelerate high-fidelity simulations:

- **Hardware-oriented acceleration:** the goal here is to ensure that existing algorithms required to execute direct numerical simulations of complex geometries are compatible with state-of-the-hart high-performance computers (HPCs). In the context of modern-day exascale computing dominated by power-efficient GPUs tailored to scientific applications [7, 302], this amounts to ensuring that algorithms are vectorized in formats compatible with the single-instruction, multiple-thread (SIMT) execution framework [16]. Put another way, hardware-oriented acceleration targets algorithm scalability and throughput optimization on modern HPC platforms, where the objective is to decrease time-to-solution by ensuring conventional CFD algorithms sustain peak-performance on the available computational nodes during the entirety of the simulation run [222].

- **Model-oriented acceleration:** in contrast to hardware-oriented acceleration, which amounts to flow solvers playing a catch-up game with the rapid evolution of hardware, model-oriented acceleration seeks to increase time-to-solution by deploying models derived from scientific modes of inquiry (experiments or simulations, as illustrated in Fig. 1.4) that are consistent with prior physical knowledge and empirical observations (data) [79, 258]. The goal of these models is to either (a) drop the total number of degrees of freedom in the simulation by reducing the number of required spatial grid points and/or required time steps, or (b) derive alternate, cheaper analytic formulations of traditionally computationally expensive algorithms by leveraging physical assumptions.

The scope of this dissertation is concerned with developing novel strategies for accelerating reacting flow simulations via the latter pathway, namely model-based

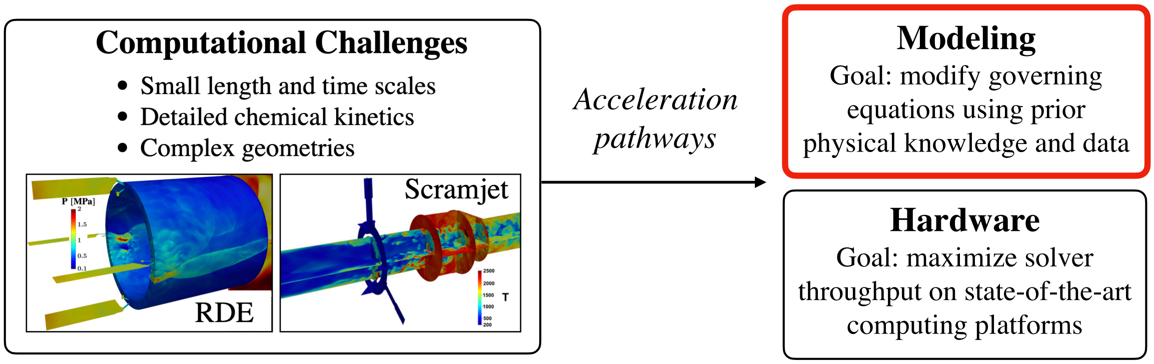

Computational Challenges
• Small length and time scales
• Detailed chemical kinetics
• Complex geometries
P [MPa]
RDE
Scramjet
T
Acceleration pathways
Modeling
Goal: modify governing equations using prior physical knowledge and data
Hardware
Goal: maximize solver throughput on state-of-the-art computing platforms

Figure 1.9: Pathways for accelerating high-fidelity, full-gometry simulations of compressible reacting flow. The scope of this dissertation is focused on model-based acceleration (boxed in red). RDE and scramjet snapshots reproduced from Ref. [30].

simulation acceleration. Within this theme, to overcome the bottlenecks described in Sec. 1.4, this dissertation explores *data-driven* strategies – the idea is to use the large amount of data accumulated from a small number of high-fidelity simulations to develop fast predictive models such that extrapolations into new designs can be explored in a much more cost-efficient way. It should be noted that although both pathways in Fig. 1.4 are presented as separate, they are very much intertwined through the idea that any developed modes should also be compatible and scalable with evolving hardware. Because of this requirement, often times hardware and HPC trends may drive the way models are constructed [130, 276] – as such, although the focus here is on model-oriented acceleration, a brief overview of hardware-oriented acceleration (particularly GPU computing) and its influence on high-fidelity reacting flow simulations is provided later on in Chapter II, Sec. 2.3.

To conclude this chapter, the following gives an overview of the remaining chapters in the dissertation. In Chapter II, the governing equations of compressible reacting flow (the Navier-Stokes equations) are presented (Sec. 2.2), an overview of HPC trends is provided (Sec. 2.3), and a detailed survey of modeling approaches used for simulation acceleration in reacting flow is presented under the classes of physics-based (Sec. 2.4) and data-based (Sec. 2.5) models. Details on the research contribution provided by this dissertation are then provided in Sec. 2.6. The contribution relies on combining beneficial qualities of both physics-based and data-based approaches in a physics-informed data-driven modeling strategy grounded in unsupervised data clustering, with primary focus on in-situ acceleration of expensive kinetics evaluations. The approach, termed classification-based regression of chemical kinetics, relies on utilizing a clustering strategy to isolate regions of dynamical similarity with reacting flowfields such that localized models can be deployed for speeding up chemical source term evaluations. Given this research contribution, Chapter III presents the fundamentals (e.g. algorithms, advantages, and limitations) of the clustering strategy used, namely K-means clustering, and provides examples of extending the baseline K-means algorithm for reduced order modeling in turbulent combustion applications for motivational purposes. Then, in Chapter IV, the classification-based regression strategy is presented and applied to detonation-containing flows. In Chapter V, the methodology for extending the clustering strategy used in Chapter IV (standard K-means) to account for underlying physical constraints from the governing equations is formulated, leading to the development of the class of physics-guided K-means clustering algorithms. Applications of the approach are demonstrated on similar detonation-containing flows as used in Chapter IV. Concluding remarks, future directions, and outlooks are provided in Chapter VI. Lastly, the Appendices contain additional information related to derivations concerning the K-means algorithm (Appendix A), matrix-based evaluation of analytic chemical source terms on GPUs (Appendix B), and GPU-optimal chemical time integration routines relevant to compressible reacting flow solvers (Appendix C).

CHAPTER II

Models for Accelerating High-Fidelity Reacting Flow Simulations

2.1 Introduction

Chapter I described the need for high-fidelity compressible reacting flow simulations for both physical analysis and design of next-generation propulsion devices, and concluded with comments on the importance of exploring acceleration pathways for these simulations to overcome the wide variety of computational challenges/bottlenecks (Sec. 1.4). As immense research has been carried out to this end in the past several decades, this chapter builds on the concluding comments of Chapter I and provides to the reader a detailed survey and classification of select modeling strategies used to accelerate compressible reacting flow solvers within the greater CFD and numerical combustion community.

Additionally, alongside surveying modeling strategies, a brief description of the hardware perspective – particularly related to the heterogeneous node architectures that dominate state-of-the-art supercomputing – will be provided to emphasize the fact that modeling frameworks compatible with emerging trends in HPC (namely GPU computing) should be preferred over alternatives.

The goal of this chapter is twofold: the first objective is to provide a detailed survey of routinely used modeling approaches used within the numerical combustion and CFD community to accelerate high-fidelity simulations for reacting flow applications, and the second objective is to present the research contribution of this dissertation in light of the shortcomings of these existing modeling approaches. In general, regardless of the class in which a given modeling approach fits, models achieve simulation acceleration using some combination of the following three overarching goals: (1) acceleration by means of spatial order reduction, (2) acceleration by means of eliminating limiting timescales (increasing the allowable simulation time step), and (3) acceleration by means of reducing arithmetic intensity of computationally intensive routines. The survey of modeling approaches below is not meant to be completely exhaustive. It is intended to describe some of the more popular approaches that are used in frameworks that not only drive faster alternatives to unsteady high-fidelity DNS of reacting flows in terms of time-to-solution, but are also inherently compatible with modern state-of-the-art high-performance computing trends.

As an overview, Sec. 2.2 presents the governing partial differential equations (PDEs) for compressible reacting flow with emphasis on the dynamical systems perspective. Then, in Sec. 2.3, to motivate the survey of modeling approaches, high-performance computing trends are presented to (a) outline the basic concepts of hardware-oriented acceleration, and (b) describe the way in which flow solvers have adapted to the now prominent heterogeneous computing paradigm. This is followed by the survey of existing modeling approaches used in reacting flow simulations, with models classified into two main categories: physics-based models (Sec. 2.4) and data-based models (Sec. 2.5). Within this classification, the ways in which these models achieve solver acceleration are discussed and their practical limitations are outlined. In light of the survey presented in Sec. 2.4 and 2.5, the chapter concludes in Sec. 2.6 with a description of the primary research contribution of this dissertation. The research contribution is the development of a physics-informed data-driven modeling strategy for accelerating chemical kinetics evaluations. The method utilizes flowfield classification grounded in data clustering, and additional emphasis in the discussion in Sec. 2.6 is placed on developing modeling frameworks amenable to in-situ, or online, parameter updates (i.e. models that adapt to the simulation as it progresses).

2.2 Governing Equations

The governing equations are the compressible reacting Navier-Stokes equations. They are given as

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0, \tag{2.1}$$

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho u_i u_j + p\delta_{ij}\right) = \frac{\partial \tau_{ij}}{\partial x_j}, \tag{2.2}$$

$$\frac{\partial \rho e_t}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho u_j e_t + p u_j\right) = \frac{\partial \tau_{ij} u_i}{\partial x_j} + \frac{\partial}{\partial x_j}\left(\kappa \frac{\partial T}{\partial x_j}\right), \tag{2.3}$$

and

$$\frac{\partial \rho Y_k}{\partial t} + \frac{\partial \rho u_j Y_k}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\rho D \frac{\partial Y_k}{\partial x_j}\right) + \dot{\omega}_k, \quad k = 1, \ldots, N_S. \tag{2.4}$$

In the above equations, repeated indices indicate summations (Einstein convention) – Eq. 2.1 is the conservation of mass, Eq. 2.2 is the conservation of momentum, Eq. 2.3 denotes the conservation of energy, and Eq. 2.4 the conservation of species concentration. The quantity ρ denotes the fluid density, u_i the fluid velocity, p the pressure, e_t the total energy, and Y_k the mass fraction for species k of N_S total species. In the above formulation of the energy equation, the total energy e_t includes chemical, sensible, and kinetic contributions, and is defined by

$$e_t = \int_{T_0}^{T} C_p dT + \sum_{k=1}^{N_S} \Delta h_{f,k}^0 Y_k + \frac{1}{2} u_i u_i - \frac{p}{\rho}, \tag{2.5}$$

where C_p is the heat capacity of the mixture at constant pressure and $\Delta h_{f,k}^0$ is the formation enthalpy of the k-th species obtained at a reference temperature T_0 [261].

The stress tensor, which contributes to the diffusion terms in Eqs. 2.2 and 2.3, is defined by

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \mu \frac{\partial u_k}{\partial x_k} \delta_{ij}. \tag{2.6}$$

The transport coefficients present in Eqs. 2.3, 2.4 and Eqs. 2.6 – the dynamic viscosity μ, the thermal conductivity κ, and the species diffusivity D – are typically estimated numerically in kinetics libraries (e.g. `Chemkin` [144] or `Cantera` [109]) by means of tabulations facilitated by high-order polynomials of temperature. Additionally, the equations are closed by assuming an ideal gas equation of state,

$$p = \frac{\rho \hat{R} T}{W}, \tag{2.7}$$

where $\hat{R}$ is the universal gas constant and W is the mean molecular weight of the mixture.

2.2.1 Chemical Source Term

The chemical source term $\dot{\omega}_k$ is present only in the conservation equation for species mass fraction because the total energy is defined in Eq. 2.5 to include the chemical contribution (species formation enthalpies). As described in Sec. 1.4, the chemical source terms are obtained using an Arrhenius formulation from detailed chemical kinetic mechanisms, where N_S is usually on the order of 10 or greater. The mechanisms are described by a collection of N_R elementary reactions, which can be concisely represented through a symbolic notation for the set of species $\{\mathcal{S}_1, \ldots, \mathcal{S}_{N_S}\}$. In this notation, a general detailed chemical mechanism is represented as

$$\sum_{k=1}^{N_S} \boldsymbol{\nu}'_{kj} \mathcal{S}_k \rightleftharpoons \sum_{k=1}^{N_S} \boldsymbol{\nu}''_{kj} \mathcal{S}_k, \quad j = 1, \ldots, N_R, \tag{2.8}$$

where $\boldsymbol{\nu}' \in \mathbb{R}^{N_S \times N_R}$ (respectively, $\boldsymbol{\nu}''$) is the reactant (respectively, product) stoichiometric coefficient matrix, and the net stoichiometric coefficient matrix is produced as $\boldsymbol{\nu} = \boldsymbol{\nu}'' - \boldsymbol{\nu}'$. The chemical source term, which is defined as a net production rate (kg/m^{3}s) for species concentration ρY_k, is a linear combination of net reaction rates:

$$\dot{\omega}_k = W \sum_{j=1}^{N_R} \boldsymbol{\nu}_{kj} Q_{net_j}. \tag{2.9}$$

In Eq. 2.9, $\mathbf{Q}_{net_j}$ contains the net reaction rate for elementary reaction j, where $j = 1, \ldots, N_R$. The characteristic nonlinearity of the chemical source term comes from the definition of the net reaction rate, which is expressed as

$$Q_{net_j} = Q_{f_j} - Q_{r_j} = K_{f_j} \prod_{k=1}^{N_S} C_k^{\nu'_{kj}} - K_{r_j} \prod_{k=1}^{N_S} C_k^{\nu''_{kj}}. \tag{2.10}$$

Above, Q_{f_j} and Q_{r_j} are the forward and reverse reaction rates for reaction j, respectively; K_{f_j} and K_{r_j} are the forward and reverse rate constants, respectively; and C_k is the species molar concentration. The forward rate constant satisfies the Arrhenius model – the reverse rate constant can either also be cast in the Arrhenius form, or, as is more commonly done, can be computed by means of evaluation of the equilibrium rate constant via NASA polynomial tabulations. The reader is referred to Ref. [109] for additional information on the numerical implementation of chemical source term evaluations using the above equations.

2.2.2 Dynamical System Formulation

To facilitate the discussion below, the PDEs presented in Eqs. 2.1-2.4 can be recast concisely through a collection of flux divergences and volumetric source terms obtained from a single conserved state vector $\mathbf{Q}(x,t) = [\rho(x,t), \rho u_i(x,t), \rho e_t(x,t), \rho Y_k(x,t)]^{\mathrm{T}} \in \mathbb{R}^{N_T}$; i indexes the spatial dimension and k indexes the N_S species, such that $N_T = 5 + N_S$ in three spatial dimensions. The compressible Navier-Stokes (CNS) equations then read as

$$\frac{\partial \mathbf{Q}(x,t)}{\partial t} + \nabla \cdot [\mathbf{A}(\mathbf{Q}(x,t)) + \mathbf{D}\mathbf{Q}(x,t)] = \mathbf{S}(\mathbf{Q}(x,t)), \tag{2.11}$$

where $\mathbf{A}$ is a nonlinear operator that represents advective fluxes, $\mathbf{D}$ is a linear operator that represents diffusive fluxes, and $\mathbf{S}$ is a nonlinear operator that represents the volumetric chemical reaction source term.

Spatial discretization of the CNS on a computational grid or mesh by means of the method of lines (e.g. utilizing finite difference [226], finite element [62], finite volume [175], or spectral methods among others [94, 243]) allows one to reformulate the governing equations in Eq. 2.11 in a semi-discrete form that invokes a dynamical system representation of the physical phenomena at play. Without loss of generality, a finite-volume based discretization is assumed to have been carried out such that the mesh is interpreted as a collection of non-intersecting N_C cells, and that the N_T conserved transport variables are stored at the centroids of each of these cells (i.e. they are cell volume averages).

In this formulation, a *dynamical system* state vector is defined at time t as $\mathbf{y}(t) \in \mathbb{R}^{N_F}$, where $N_F = N_C \times N_T$. In contrast to $\mathbf{Q}$, the state vector $\mathbf{y}$ encapsulates the instantaneous position of all transport variables stored at the cell centers in the N_F-dimensional phase space. It should be noted that in high-fidelity simulations, the quantity N_F can reach extremely high values – for example, even for a modest mesh size of 10 million cells, a reacting flow simulation utilizing $N_S = 100$ species in the detailed chemical mechanism produces a state vector comprised of over 1 billion degrees-of-freedom.

Proceeding by means of the spatial discretization method, the evolution of the dynamical system state vector is then given by

$$\frac{d\mathbf{y}}{dt} = \mathcal{A}(\mathbf{y}) + \mathcal{D}\mathbf{y} + \mathcal{R}(\mathbf{y}) = \mathbf{F}(\mathbf{y}) \in \mathbb{R}^{N_F}, \quad \mathbf{y}(t=0) = \mathbf{y}_0. \tag{2.12}$$

The above equation is characterized by (1) the nonlinear operator $\mathcal{A}$, which represents the action of computing numerical divergences of the advective fluxes using stencils that adapt to the flowfield (upwind schemes), (2) the symmetric linear operator $\mathcal{D}$, which represents the action of computing numerical divergences of diffusive fluxes (equivalent to the numerical Laplacian operator), and (3) the nonlinear operator $\mathcal{R}$, which represents the Arrhenius-based evaluations of chemical source terms for the N_F discretization points contained in the state vector $\mathbf{y}$.

2.3 High-Performance Computing Trends

Before discussing the various modeling strategies for achieving simulation acceleration, it is important to first outline key trends in the development of modern hardware and state-of-the-art high-performance computers (HPCs) as well as the way in which the CNS equations are solved on these HPCs. These trends not only give context to how flow solvers should be developed, but also govern the way in which candidate models are selected for simulation acceleration purposes. Ideally, models should be compatible with the way modern hardware has evolved to optimally execute algorithms and minimize time-to-solution. As such, this section introduces ideas of hardware-oriented acceleration (see Fig. 1.9 and associated discussion).

In broad terms, quantitative assessment of hardware capability relevant to scientific computing is derived from (a) peak performance measurements in terms of floating point operations (FLOPs) per second, which determines the rate at which the particular piece of hardware can perform arithmetic evaluations, (b) bandwidth measurements in terms of bytes-per-second, which determines the rate at which data can be transferred to and from global memory, and (c) the power consumption of the device at sustained operation, which provides a measure for energy efficiency. Although other performance-related quantities are also important [244], requirements based on the the above three quantities are the main drivers for hardware selection in HPC nodes.

With these quantities of interest in mind, trends in both peak performance and bandwidth for a subset of hardware devices are shown in Fig. 2.1 to emphasize the difference between conventional central processing units (CPUs) and the now prominent graphics processing units (GPUs). More specifically, the figure compares evolution in terms of performance (FLOPs-per-second) and bandwidth (Gigabytes-per-second) for select Intel CPUs and NVIDIA GPUs – although only a small subset is devices is shown, the implications of Fig. 2.1 are reflective of overall hardware trends that drive modern HPC for computational science [7]. Ultimately, Fig. 2.1 shows how both peak performance and bandwidth rates for state-of-the-art GPUs are steadily higher than those of CPUs – the implication is that, for scientific applications requiring high arithmetic intensity, GPUs must be utilized in some way due to their sheer computational advantage. Additionally, although not shown in Fig. 2.1, GPUs also provide greater power efficiency over CPUs in terms of FLOPs/Watt, which further motivates their usage at large scales.

The consequence is that modeling frameworks and algorithms inherently incompatible with GPUs, despite how potent they are from a mathematical or philosophical lens, are usually not considered in the computational science community if scalability is requirement. This effect is formally known as the hardware lottery [129], which ar-

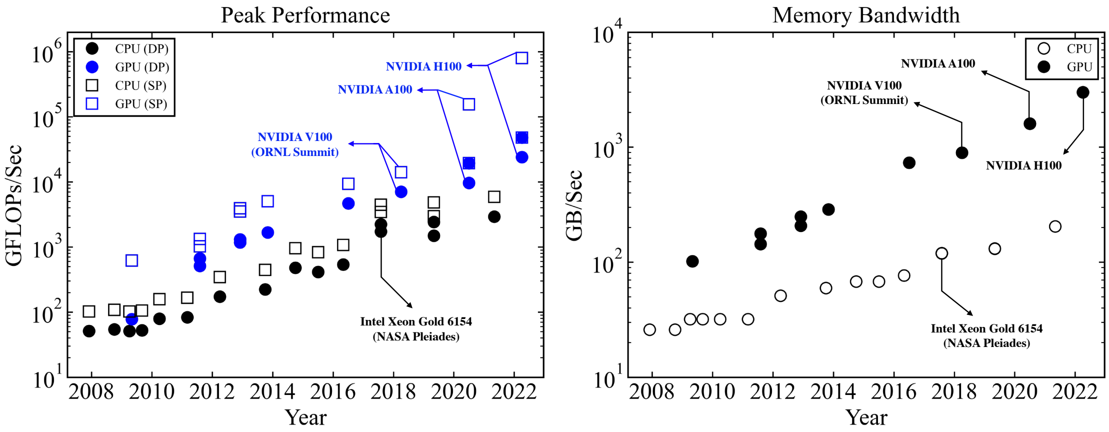
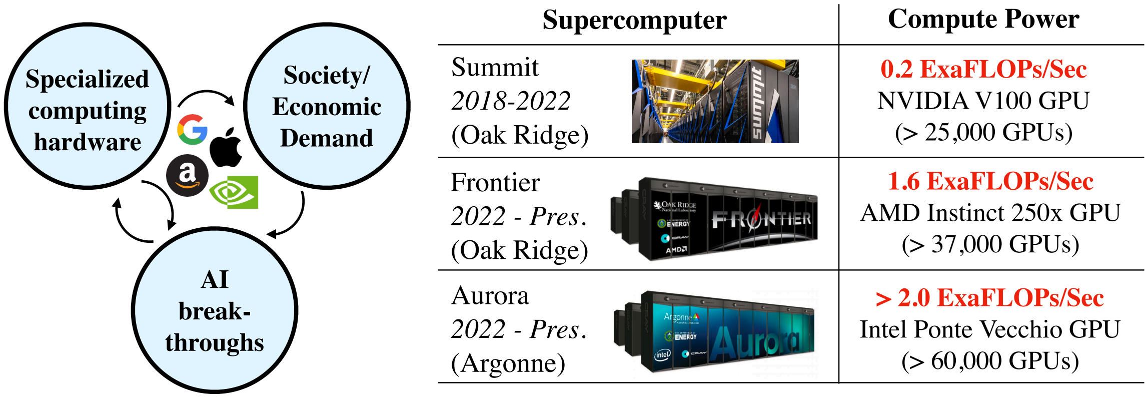


Figure 2.1: **(Left)** Evolution of theoretical peak performances for Intel Xeon CPUs (black) and NVIDIA GPUs (blue) in both single precision (empty squares) and double precision (filled circles) computing formats from 2008 to present-day. **(Right)** Evolution of memory bandwidth limitations for NVIDIA GPUs (filled circles) and Intel CPUs (empty circles). Annotations provided for hardware used in NASA Pleiades (CPU-based) and ORNL Summit (GPU-based) HPCs. Data extracted from Intel and NVIDIA specification sheets.

gues that despite the physical and mathematical viability of a wide variety of modeling frameworks in any given field of study, the tight feedback between societal demands and hardware availability at a particular time biases the scientific community towards models whose algorithms are intrinsically compatible with current hardware. As these algorithms and models become widely adopted and successful, the hardware in turn becomes more optimized for these models, which amplifies the feedback mechanism. This phenomenon, illustrated in Fig. 2.2 (left), explains the meteoric rise in the rate of adoption of methods derived from AI (e.g. neural networks) into other seemingly unrelated fields like fluid dynamics [79] and numerical combustion [133] in the 21st century. For example, the backbones of neural networks are batched matrix multiplications and convolutions which thrive in the single-instruction multiple-thread (SIMT) computing formats favored by GPUs [235]. The feedback comes from the fact that GPUs are now becoming increasingly tuned for low and mixed-precision arithmetic (visualized in the peak performance trends in Fig. 2.1) due to the societal demand for reducing costs in training and inference algorithms for very large neural



Supercomputer	Compute Power
Summit *2018-2022* (Oak Ridge)	**0.2 ExaFLOPs/Sec** NVIDIA V100 GPU (> 25,000 GPUs)
Frontier *2022 - Pres.* (Oak Ridge)	**1.6 ExaFLOPs/Sec** AMD Instinct 250x GPU (> 37,000 GPUs)
Aurora *2022 - Pres.* (Argonne)	**> 2.0 ExaFLOPs/Sec** Intel Ponte Vecchio GPU (> 60,000 GPUs)

Figure 2.2: **(Left)** Illustration of the coupling between research advances in AI and data science, development of specialized hardware (i.e. GPUs), and societal demands. **(Right)** State-of-the-art HPCs commissioned by the USA Department of Energy. Underneath compute power (red text), type of GPU used and total number of available GPUs is provided. Data and images obtained from Refs. [12, 227, 228].

networks that excel in low-precision numerical environments.

From the scientific computing perspective, these feedback mechanisms and associated hardware trends in Fig. 2.1 drive the way in which state-of-the-art HPCs are now commissioned and deployed for scientific computing applications like combustion simulations [7]. Some examples of supercomputing platforms sponsored by the USA Department of Energy are shown in Fig. 2.2 (right). With the goal of ushering in the exascale computing era, the most powerful HPCs are now entirely built on so-called heterogeneous node architectures that contain both CPUs and a set of GPU "accelerators" or "offloaders" – as the names imply, the intended workflow is to utilize the CPUs to send computationally intensive tasks to the GPUs [207]. A schematic of a heterogeneous node architecture is shown in Fig. 2.3. The fastest and most power-efficient HPCs (three of which are shown in Fig. 2.2) now consist of thousands of such heterogeneous nodes, which translates to housing on the order of tens of thousands of GPUs per HPC. In all of the HPCs listed in Fig. 2.2, the GPUs are responsible for over 95% of the overall compute power. As such, to realize hardware-oriented acceleration (Fig. 1.9), top-of-the-line flow solvers must adapt to the new heterogeneous computing paradigm by demonstrating scaling on thousands of GPUs that are

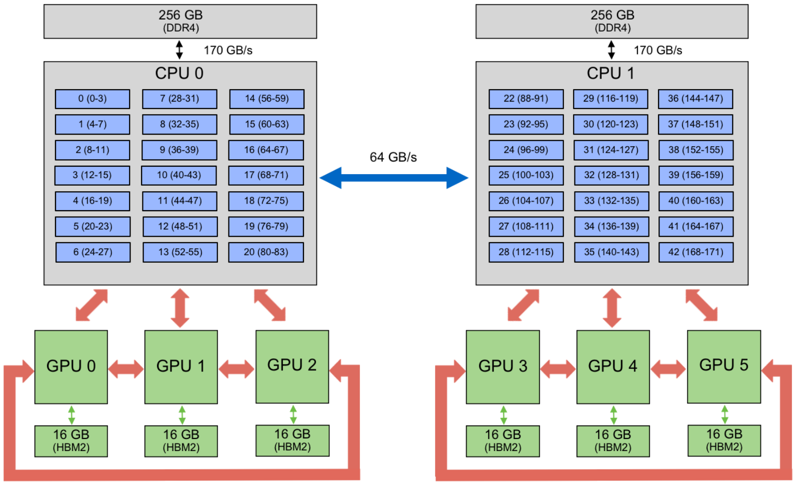


Figure 2.3: Illustration of heterogeneous node architecture used in Summit HPC. Gray boxes denote IBM Power9 CPUs, green boxes denote NVIDIA V100 GPUs, and arrows denote data transfer pathways. The GPUs are responsible for roughly 98% of the node compute power. Reproduced from Ref. [241].

optimized for AI and not CFD [30, 340].

To this end, standard methods of GPU-integration for compressible reacting flow solvers accommodates spatial discretization based on domain decomposition, shown in Fig. 2.4. In domain decomposition, a parallel programming library (usually MPI [346]) is used to assign non-overlapping subsets of the spatial domain to individual CPU cores. Each core (or MPI rank) then operates on its own subset of cells in isolation until a synchronization stage is reached, at which consistency in the flow solution along the boundaries of the subdomains is enforced in a bandwidth-limited communication operation. The presence of GPUs as the source of dominant computing power within the HPC node now requires traditional domain-decomposition based flow solvers to operate within the MPI+X framework for optimal throughput, where the ”X” refers to some type or brand of GPU to which the CPU core offloads its computationally intensive tasks. As a result, new programming frameworks that supplement standard GPU libraries like CUDA have emerged to alleviate solver development hurdles related to memory management and coding challenges in the heterogeneous MPI+X environment (examples include Kokkos [81], Raja [25], Le-

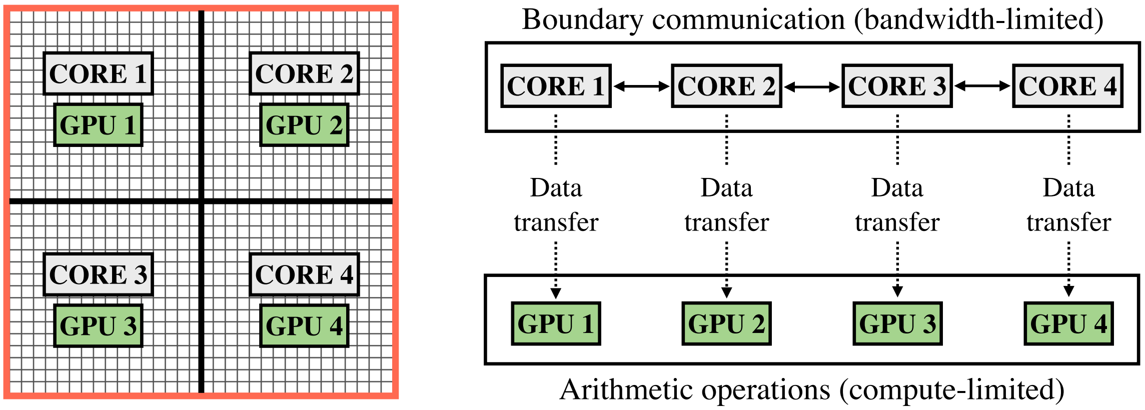


Figure 2.4: **(Left)** Illustration of domain decomposition approach for a 2D square geometry. Red lines indicates domain boundary, black lines indicate subdomain overlap regions. **(Right)** Offloading procedure between the CPU cores and GPU accelerators within the MPI+X paradigm. For illustrative purposes, this figure assumes a one-to-one mapping between core and GPU; this does not have to be the case in practice.

gion [24], and compiler directive-based frameworks like OpenMP [172] and OpenACC [352]).

All aspects of solver performance now revolve around how well the GPUs within the HPC nodes are utilized in the flow solver. Achieving theoretical peak performance on GPUs requires algorithms to operate at high levels of arithmetic intensity, defined as the ratio of floating point operations (FLOPs) determined by the algorithm arithmetic (i.e. number of multiplications and additions) to the amount of data transferred to and from the global memory source as determined by the required inputs and outputs. Pathways to achieve high arithmetic intensity are provided by rewriting (or refactoring) computationally intensive solver routines in a matrix-based or tensor format for which the GPUs are inherently designed (i.e. rewriting routines in flow solvers to mimick neural network forward passes) [16]. Aside from refactoring algorithms, other pathways for increasing arithmetic intensity include increasing the number of local cells offloaded to the GPU by the MPI rank within the domain decomposition framework until the saturation point of the GPU is reached. This results in GPU scaling behavior that may seem counterintuitive: in practice, better utilization of the GPU often comes from dropping the number of subdomains for a given geometry in the domain decomposition framework until a saturation point is reached [30].

The structure for a reacting flow solver used for numerical simulation of the compressible Navier-Stokes equations is shown in Fig. 2.5. In a fully offloaded solver, the role of the GPU is to perform all intensive arithmetic operations within the time step such that (1) GPU utilization is sustained for as long as possible, and (2) CPU-to-GPU data transfer overhead is minimized. This amounts to GPU treatment of flux and divergence evaluations for advection and diffusion terms, as well as chemical source term evaluations for reactions [30]. Note that reaction source terms for each cell can either be computed from instantaneous rate evaluations via the Arrhenius formulation (Eq. 2.9), or extracted from the result of a stiff time integrator within an operator splitting framework (see Sec. 2.4.6 and Appendix C). Offloading the computationally intensive evaluations from a CPU solver to a GPU in this manner not only results significant time-to-solution speedups, ranging from 2X to 10X in practice [30, 340], but also ensures that solvers are future-proof in the event that HPCs continue to prioritize GPU computing.

Despite the computational gains provided by GPU computing and hardware-oriented acceleration, there are a few caveats: (1) the act of offloading a CPU-based solver to perform expensive evaluations on GPUs provides a one-time speedup (once a solver is offloaded, it is "caught-up" with the state-of-the-art hardware), and (2) GPU offloading eventually reaches a restrictive MPI-derived communication bottleneck that stems from the domain-decomposition approach used to solve the governing equations – this comes from the bandwidth-limited MPI cost of communicating boundary conditions across subdomains (shown in Fig. 2.5). For these reasons, modeling approaches are required to truly realize the long-time simulation capability required for robust design strategies (see Sec. 1.5). As described next in Sec. 2.4 and

2.5, the broad goal for these models (or model-oriented acceleration) is to extract further computational gain on top of that provided by hardware-oriented acceleration to eliminate physics-derived restrictions based on high resolution requirements, such as the detailed chemistry bottlenecks mentioned in Sec. 1.4.

As a final note, some may argue the above hardware advantages provided by GPU computing have eliminated the need for highly intrusive models whose goals are to produce drastic reduction in overall degrees-of-freedom. For example, innovations in ROM frameworks in the 80s and 90s (i.e. proper orthogonal decomposition and large-eddy simulation, discussed in detail in the next sections) were driven by the disparity between desired simulation run times and available computing power. With HPCs now reaching the exascale (see Fig. 2.2), the argument for motivating model development due to lack of computing power no longer holds, which limits the traditional scope of ROMs. In other words, ensuring ROM compatibility with HPCs requires the amount of order reduction provided for a single simulation to decrease, either by (a) retaining more modes in the context of modal decomposition based ROMs, or (b) increasing the grid resolution in implicit large-eddy simulations. These qualities effectively inch ROM frameworks closer to the capabilities of direct numerical simulations (DNS), to the point where it may be more worthwhile to concentrate research effort towards accelerating solver routines using non-intrusive models embedded in DNS, as opposed to completely replacing DNS with ROMs. This quality has motivated the research contribution of this dissertation, which as outlined in Sec. 2.6, involves accelerating prohibitive chemical kinetics routines found in reacting flow solvers.

2.4 Physics-based Models

Physics-based models utilize the governing equations as the starting point for model development, either from the PDE perspective (Eq. 2.11) or the semi-discrete dynamical systems perspective (Eq. 2.12). Detailed descriptions of some widely used

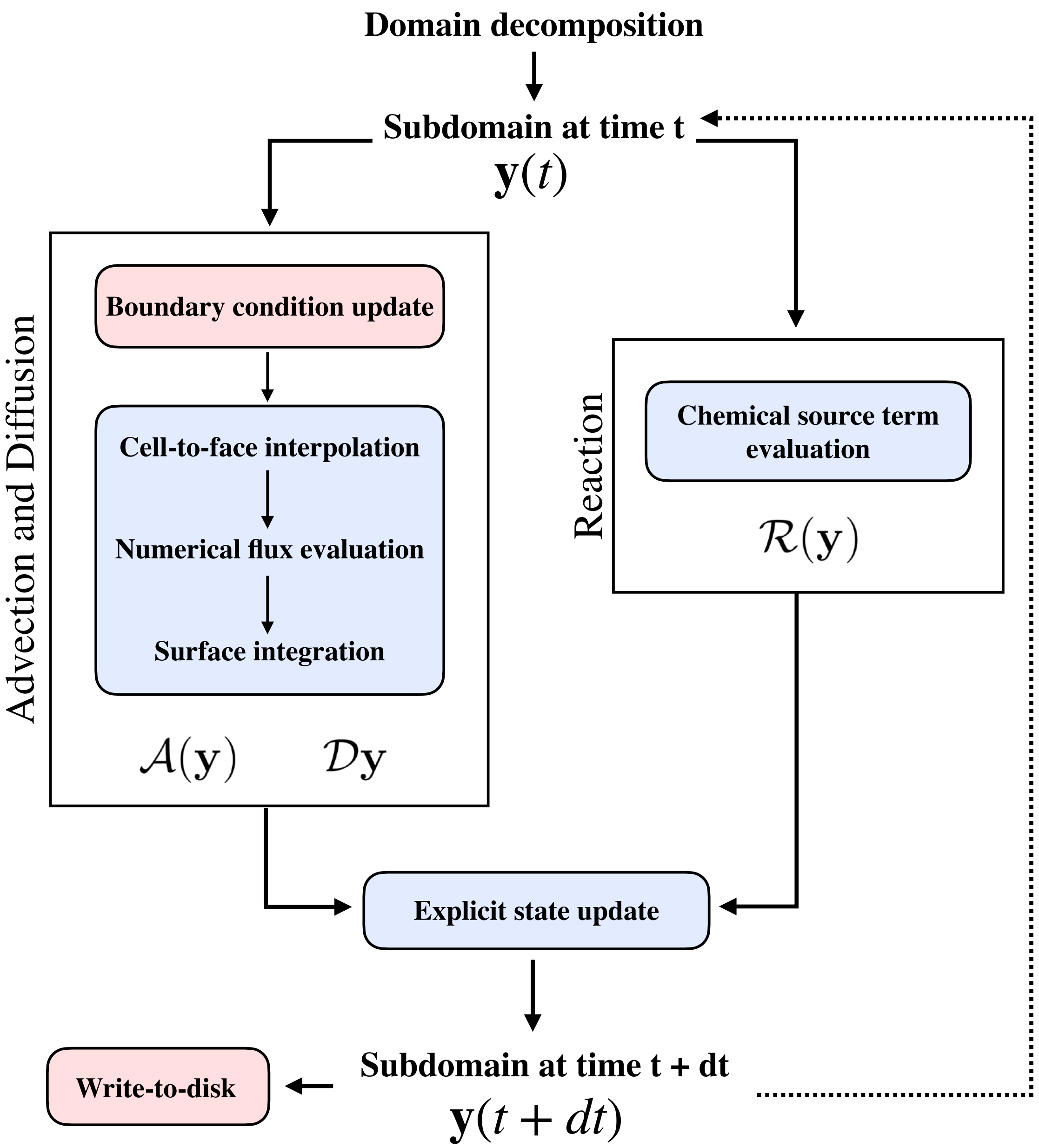


Figure 2.5: Flowchart for GPU-based finite-volume explicit compressible reacting flow solver (based on UMReactingFlow [30]). Red boxes indicate CPU-only evaluations (the bottlenecks). Blue boxes indicate areas that should be offloaded to GPU. Flowchart operates under domain decomposition framework of Fig. 2.4 and dynamical system of Eq. 2.12.

methods in reacting flow simulations are provided below – additional reviews of methods used in reacting flow applications are provided in Refs. [258, 261, 276]. It should be noted that the methods discussed below are relevant for accelerating predictions of unsteady reacting flow phenomena. As such, low-fidelity statistical frameworks that produce time-averaged flowfields like Reynolds-averaged Navier-Stokes [245] are not discussed.

2.4.1 Large-Eddy Simulation

Large-eddy simulation (LES), the modern versions of which are built on the initial studies by Smagorinsky [318], is a statistical reduced-order modeling (ROM) approach for turbulent flows [267]. Significant research effort in the past decades has put LES at the forefront for the accelerated prediction of unsteady combustion and turbulent reacting flow phenomena [258], from canonical free-shear applications like turbulent reacting jets [142, 277] and jets in crossflow [143], to full-geometry simulations of industrial propulsion devices [168, 238, 291].

LES takes advantage of the concept of scale separation and energy cascade in canonical turbulent flows. The large scales that contain a majority of the turbulent kinetic energy are directly resolved in the simulation and the small scales are modeled [267]. The ideal cutoff between large and small scales is an expert-guided input, and is determined not only by the physics of the problem through a representative turbulent Reynolds number, but also by target prediction horizon times for a given application and the level of desired reduction in the total number of degrees-of-freedom.

In LES, the governing equations in Eq. 2.11 are passed through spatial filter of size Δ, producing a set of *filtered equations* that are solved numerically (Ref. [267], Chapter 13). The filtering step acts as a scale separation mechanism through the filter size Δ: lengthscales above Δ are called resolved scales and are represented in the numerical solution, whereas lengthscales below Δ are called unresolved scales.

Since only the resolved scales are represented in the filtered equations, the modeling goal in LES is to properly account for the dynamical impact of the unresolved scales on the resolved scales without directly representing the unresolved scales numerically. Because the governing equations are nonlinear, this constitutes an elusive closure problem that is referred to as the sub-grid scale modeling task. Higher filter widths produce greater potential for speedup by means of reducing the overall number of degrees-of-freedom in the underlying dynamical system (Eq. 2.12), but also increase the impact of the unresolved scales on the overall dynamics, which in turn implies a more difficult role for the closure modeling task.

The starting point in LES is to define a filter. The filter can be defined implicitly or explicitly – implicit filtering, which is the most commonly used approach, interprets the mesh as the output of a filtering approach. In implicit LES, the user does not directly execute a filtering operation in the code; rather, the starting point is to set an under-resolved characteristic lengthscale for the cell size, and proceed with the simulation on this coarse mesh [196]. On the other hand, in explicit LES (or explicit filtering), the unresolved scales are recovered in an approximate manner by means of algebraic relations based on the filter definition [186] – this can be accomplished with methods like approximate deconvolutions [197] or digital filtering [153].

Regardless of which filtering method is used, the end-result is that it must result in information loss by means of a reduction in the dimensionality, or degrees-of-freedom, of the dynamical system in Eq. 2.12. The filter must also be non-invertible. If this property is not satisfied, the dynamics of the unfiltered equations, which are the high-fidelity direct numerical simulations, would be identical to the filtered equations, resulting in a trivial ROM.

There are many pathways by which one can mathematically present the central concepts of large-eddy simulations. Here, we will opt for the description more in-line with the dynamical systems representation provided in Eq. 2.12. This formulation,

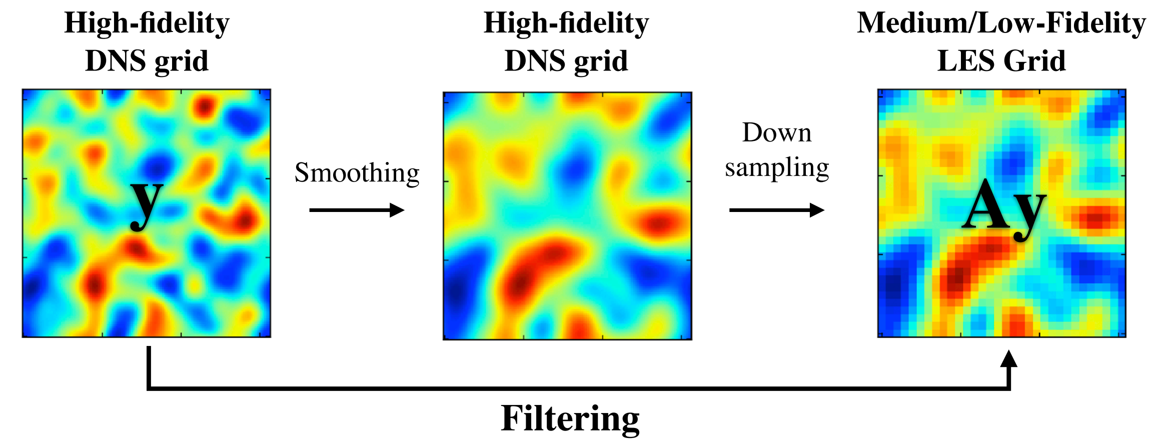


Figure 2.6: Interpretation of the effect of filtering in large-eddy simulations via non-invertible linear operator $\mathbf{A}$ [165].

known as ideal LES [165], is derived from methods of stochastic estimation and probability density function (PDF) transport, as described in the works of Adrian [2] and Pope [266]. As a starting point, we define the filter as a linear, non-invertible operator $\mathbf{A}$. The action of $\mathbf{A}$ on the fully resolved high-fidelity flowfield $\mathbf{y} \in \mathbb{R}^{N_F}$ produces a filtered flowfield $\hat{\mathbf{y}}$ that not only eliminates the unresolved scales below the filter width prescribed by $\mathbf{A}$, but also drops the number of degrees-of-freedom of $\mathbf{y}$ from N_F to N_F^{LES}. That is,

$$\hat{\mathbf{y}} = \mathbf{A}\mathbf{y} \in \mathbb{R}^{N_F^{LES}}. \tag{2.13}$$

Intuitively, the filter procedure in Eq. 2.13 can be described as the result of two successive steps: (1) a smoothing step that eliminates the lengthscales below the filter width from the ground-truth instantaneous flowfield $\mathbf{y}$, and (2) a down-sampling step that reduces the dimensionality of the flowfield and effectively moves it to the LES grid. An example of the effect of the filtering operation is shown in Fig. 2.6.

The governing equation for the filtered high-fidelity flowfield (filtered DNS) can then be obtained from Eq. 2.12 as

$$\mathbf{A}\frac{d\mathbf{y}}{dt} = \mathbf{A}\mathbf{F}(\mathbf{y}) \in \mathbb{R}^{N_F^{LES}}. \tag{2.14}$$

In an LES simulation, the high-fidelity DNS fields given by $\mathbf{y}$ are never available. Instead, the LES procedure solves for a different flowfield, denoted $\mathbf{w}$, which exists on the LES grid. The evolution of the LES field is defined by the dynamical system

$$\frac{d\mathbf{w}}{dt} = \mathbf{F}(\mathbf{w}) + \mathbf{M}(\mathbf{w}) \in \mathbb{R}^{N_F^{LES}}, \tag{2.15}$$

where $\mathbf{F}(\mathbf{w})$ denotes the evaluation of the discretized compressible reacting Navier-Stokes equations on the LES grid, and the quantity $\mathbf{M}(\mathbf{w})$ denotes the LES model. Note that both Eqs. 2.14 and 2.15 exist on the same mesh, which is designed to not resolve all relevant lengthscales for the problem at hand (see Sec. 1.4 for a discussion on lengthscale requirements). As described in Ref. [165], the LES model can *at best* be equivalent to the following residual,

$$\mathbf{M}(\mathbf{w}) = \left\langle \mathbf{A}\frac{d\mathbf{y}}{dt}\middle|\mathbf{A}\mathbf{y} = \mathbf{w}\right\rangle - \mathbf{F}(\mathbf{w}), \tag{2.16}$$

where the term in brackets denotes an ensemble average of the filtered RHS conditioned on all possible DNS (or high-fidelity) flowfields that produce the instantaneous LES field $\mathbf{w}$ when filtered.

Since the conditional average in Eq. 2.16 cannot be feasibly computed (it requires evaluating companion ensemble DNS simulations), we instead use conventional LES models with extensions to reacting flow based on eddy viscosities and diffusivities for predicting subgrid stress tensors, scalar variances, and scalar dissipation rates. The model-form errors in these ad-hoc LES models should be compared with the residual defined by the ideal formulation in Eq. 2.16. Overall, the advantage of LES models is their simplicity and compatiblity with HPC trends: frameworks that utilize LES can increase throughput by simply decreasing the filter width. However, despite the widespread usage of eddy-viscosity based LES models in turbulent combustion applications [43, 105, 258, 259], these models are derived under simplistic canonical configurations based on local homogeneity and isotropy of the subgrid scales, which are violated in complex, advection-dominated environments found in propulsion devices like rotating detonation engines and scramjets [278]. Additionally, the problems of correctly prescribing filter widths, grid convergence trends, and wall models in LES applications remain either unsolved for complex reacting flows or highly configuration-specific [36, 265]. Despite this, because of demonstrated success on key applications (particularly in the low-Mach regime), LES continues to remain a go-to tool for simulation of all types of turbulent fluid flows at various levels of fidelity.

2.4.2 Combustion Models

In a broad sense, combustion models of various forms use principles derived from canonical flame configurations to achieve simulation acceleration by (a) removing the need for detailed chemical kinetic mechanisms and (b) ensuring compatibility with large-eddy simulations [258]. These methods have seen significant success in several reacting flow applications primarily in the low-Mach regime, such as simulations for soot formation in gas turbines [58, 116, 213], prediction of complex ignition phenomena [48, 331], simulations of jet-in-crossflow configurations [351], and representation of multi-phase combustion [155].

Adopting the notation of Ref. [276], a general combustion model takes the form

$$\phi = \mathcal{G}(\psi) \in \mathbb{R}^{N_\phi}, \tag{2.17}$$

where the variable ϕ is a chemical composition vector of size N_ϕ, ψ is an input vector of size N_ψ, and $\mathcal{G}$ is the combustion model that can be interpreted as a mapping function. Within the LES framework, combustion models are used primarily to supply filtered scalar fields through the input variable ψ as

$$\widetilde{\phi} = \int \mathcal{G}(\psi)\rho(\psi)d\psi, \tag{2.18}$$

where $\widetilde{\phi}$ denotes the filtered composition vector and $\rho(\psi)$ the joint sub-filter PDF of the input variables. Note that the formulation in Eq. 2.18 generalizes to non-LES settings (e.g. in situations where the grid resolution is below all required lengthscales) if the joint-PDF is supplied as a delta function. Combustion modeling research efforts can be divided into four major focuses [276]: (1) reducing the size of the composition vector ϕ (this amounts to replacing detailed chemistry with reduced chemistry formulations), (2) selecting the proper input variables ψ for the task at hand, (3) derivation of the combustion model $\mathcal{G}$ from canonical configurations, and (4) evaluation and parametrization of the joint PDF ρ, which is nontrivial in turbulent combustion applications because it is a function of both space and time. Based on the method used, the amount of complexity attributed each of these four thrusts is different. For example, PDF transport approaches solve for the joint PDF ρ directly using a separate set of PDEs and/or particle-in-cell methods, while representing $\mathcal{G}$ as an identity map [264, 277]. On the other hand, mechanism reduction [182] and manifold-based approaches [213] (one of which is discussed below) reduce the dimensionality of ϕ, such that developing the combustion model $\mathcal{G}$ becomes much more computationally feasible.

One modeling framework that has achieved significant success in the combustion community is the flamelet/progress variable (FPV) method. This method was initially introduced by Pierce and Moin [255, 256] for diffusion flames and has since seen continued development into premixed [208, 331] and partially-premixed configurations [220, 269]. As the name implies, FPV methods model turbulent flames as a collection of laminar flamelets [251], which are loosely defined as regions in which chemical reactions occur in locally stoichiometric regions at timescales smaller than those of turbulence. Based on the physical characteristics of flamelets, FPV approaches design the combustion model $\mathcal{G}$ to recover solutions from a set of PDEs called the flamelet equations. The derivation of these equations depends entirely on the type of combustion processes one wishes to capture. In many applications, treatment of diffusion flames utilize steady-state solutions to the flamelet equations [258], whereas treatment of premixed flames often retain the unsteady terms [188] and derive the flamelet PDEs from either simplified counterflow or freely-propagating flames (as shown in Fig. 2.7). Based on the canonical configuration used, many FPV methods define ψ using two major input parameters: (1) the mixture fraction Z, which can be interpreted as a normalized equivalence ratio [252], and (2) the progress variable C, which is a problem-dependent linear combination of a small subset of the total set of N_S species mass fractions.

The FPV workflow is as follows. In a pre-processing step, the flamelet equations are solved numerically with no models using detailed chemistry, which effectively samples the true joint-PDF of mixture fraction and progress variable. Then, the combustion model $\mathcal{G}$ is constructed using a tabulation strategy, where the input vector ψ includes Z, C, and other variables depending on the application [217]. The combustion model $\mathcal{G}$ is then deployed in the flow solver (usually LES-based): LES generates the required inputs ψ for the combustion model (updated mixture fraction, progress variable, and other required parameters like scalar dissipation rate), and the combustion model generates the filtered scalar fields required for LES [258]. In general, FPV accelerates high-fidelity reacting flow simulations based on detailed kinetics by reducing the size of the composition vector N_ϕ from N_S (the number of species) to 2 (mixture fraction and progress variable). This reduction comes from the invocation of a flamelet-generated manifold (FGM) [220, 343]. In other words, instead of transporting mass fractions for all species as described by Eq. 2.4, the

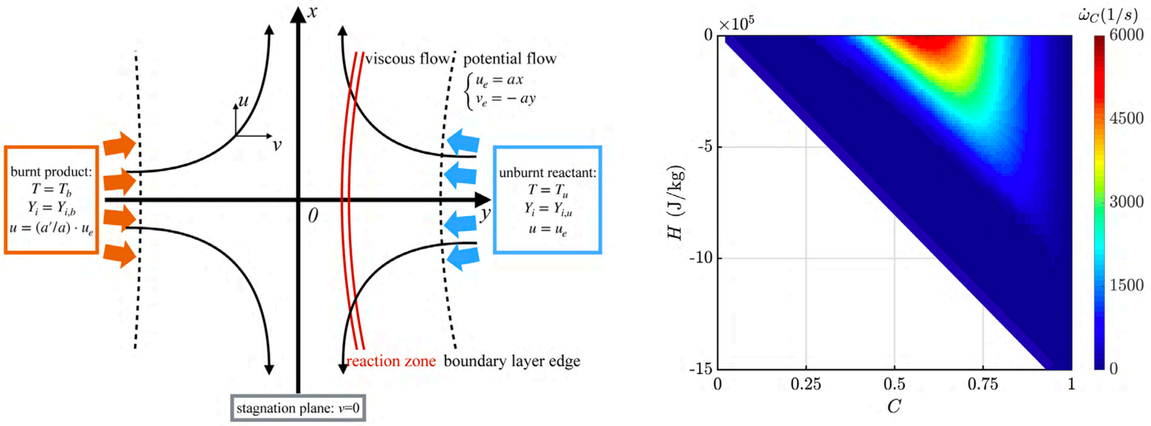


Figure 2.7: **(Left)** Example of a counterflow flame configuration used to derive the flamelet equations. **(Right)** Output combustion model $\mathcal{G}$, here shown as a tabulation of progress variable source term in the enthalpy (y-axis) and progress variable (x-axis) space. Both figures reproduced from Ref. [331].

FPV-based flow solver instead transports only mixture fraction and progress variable. As shown in Fig. 2.7, the combustion model produces the effect of the chemical source term through a tabulation strategy without having to evaluate the expensive Arrhenius-based expression associated with detailed chemistry in Eq. 2.9.

Despite the widespread usage and modeling capabilities provided by FPV approaches, they have significant limitations [155]. Perhaps the biggest lies in the selection of the canonical regime used to derive the flamelet equations. Conventional derivations used to produce flamelet models assume constant pressure, which are valid in low-Mach applications but break down in scenarios described by compressible reacting flow required in simulations of scramjets and RDEs. Although methods that extend compressibility into FPV methods have been recently explored [290, 292], they add additional costs and assumptions when solving the unsteady flamelet equations used to produce the combustion model, and their extension into complex configurations has yet to be demonstrated. Additional limitations are attributed to proper specification of the input variables ψ (i.e. what to include alongside Z and C) – accounting for complex physical effects such as heat loss, strain, ignition, as well as compressibility, requires appending ψ with several additional variables that may render accurate tabulation of the flamelet PDE solutions infeasible. Another major challenge for combustion modeling in general is the specification of the joint PDF of the input variables – to avoid transporting this PDF in the solver directly, it is commonplace to enforce simplified expressions (e.g. beta PDFs [347]) to keep computational costs small. Other combustion models, such as the level-set based G-equation method used to treat premixed turbulent combustion [253], have similar limitations – exhaustive discussion on other combustion modeling methods is provided in Refs. [258, 276] and the references therein.

2.4.3 Adaptive Mesh Refinement

Adaptive mesh refinement (AMR) is another physics-based modeling approach that has connections with nonlinear multigrid methods [39, 138]. Early applications of AMR on compressible flow were performed by *Berger and Oliger* [28] for non-reacting shock-containing solutions of the Euler equations – since then, various formulations of AMR have been popularized and extended to several complex reacting flow applications in both high-speed [70] and low-Mach regimes [248].

In AMR, a baseline mesh is set to an under-resolved level. Using a set of local error estimators that flag regions in the flowfield which observe high levels of numerical discretization error, the baseline mesh (called the coarse grid) is refined such that acceptable high-fidelity resolution requirements for the simulation are satisfied in a subset of the domain instead of in the entire domain. If the error estimators (usually flow gradients computed on the coarse grid [344]) are highly localized in space, the hope is that the AMR approach produces the same results as a single-grid high-fidelity simulaton with a significantly reduced total number of spatial discretization points or degrees-of-freedom.

As with LES, there are many variants of AMR depending on whether or not structured grids, unstructured grids, or overset meshes are utilized [260]. Regardless of the variant used, a key feature of AMR is that the mesh changes with the potentially unsteady local flow features that are tagged for refinement (e.g. traveling shockwaves, detonation waves, or free shear layers). The requirement of mesh adaptation invokes an interpolation step from a coarse grid to a fine grid in regions where the error estimators activate in places where no fine grids existed previously. This interpolation step is inherently ill-defined and is similar in spirit to explicit filtering methods used in LES.

Without loss of generality, the discussion here focuses on one such variant known as block-structured AMR developed by *Berger and Colella* [27]. Here, the baseline coarse grid is nested with successively finer grid levels in high-error regions until the desired resolution is reached. So long as grid nesting guidelines are satisfied, an arbitrarily high number of refinement levels can be added. A schematic of the meshing approach is shown in Fig. 2.8. Grids on different levels are treated as independently evolving simulations; the flowfields on fine grid levels are then ”synced” with the overlapping solutions on coarse grid cells based on the simulation time step by means of a volume averaging procedure. For coarse cells tagged by the error estimators, coarse-to-fine interpolation routines provide initial and boundary conditions for finer grid levels.

In the AMR framework, the evolution equation on the coarse grid level has the form

$$\frac{d\mathbf{y}_c}{dt} = \mathbf{F}(\mathbf{y}_c) + \mathbf{M}(\mathbf{y}_c) \in \mathbb{R}^{N_F^{coarse}}, \quad \mathbf{y}_c(t=0) = \mathbf{y}_{c,0}. \tag{2.19}$$

where $\mathbf{F}(\mathbf{y}_c)$ is the same forcing obtained from numerical discretization of the compressible Navier-Stokes equations as in Eq. 2.12, and $\mathbf{M}$ is a residual forcing term that is assumed to be significant in regions where high flow gradients exist. The fine grid equations are given by

$$\frac{d\mathbf{y}_f}{dt} = \mathbf{F}(\mathbf{y}_f) \in \mathbb{R}^{N_F^{fine}}, \quad \mathbf{y}_f(t=0) = \mathcal{I}_c^f[\mathbf{y}_{c,0}], \tag{2.20}$$

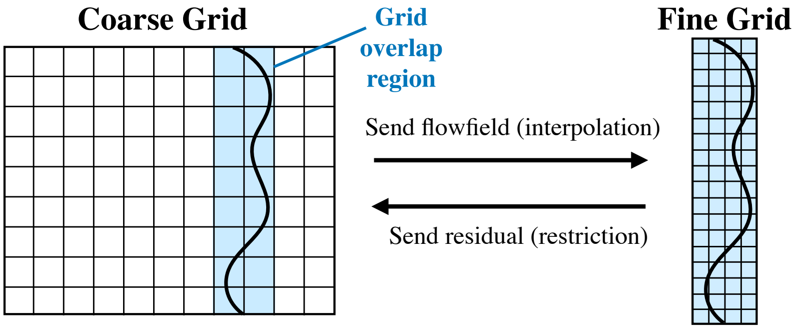


Figure 2.8: Schematic of coarse grid and fine grid communication steps carried out in block-structured AMR. Grid overlap region (blue) ideally contains a propagating discontinuity.

where $\mathcal{I}_c^f$ denotes a coarse-to-fine interpolation operation. Equation 2.20 is mathematically identical to Eq. 2.12 – the only difference is the number of degrees-of-freedom, N_F^{fine}, which should be significantly smaller than the single-grid high-fidelity dimensionality N_F for AMR to be cost-effective. The residual forcing function $\mathbf{M}$ on the coarse grid, which constitutes the modeling approach in AMR, is obtained from fine grid projections as

$$\mathbf{M}(\mathbf{y}_c) = \mathcal{I}_f^c[\mathbf{F}(\mathbf{y}_f)] - \mathbf{F}(\mathcal{I}_f^c[\mathbf{y}_f]), \tag{2.21}$$

where $\mathcal{I}_f^c$ denotes a conservative fine-to-coarse restriction operation. The procedure in Eq. 2.21 can be interpreted as a residual correction analogous to nonlinear multigrid methods (termed the full approximation scheme correction [82]) or as the result of a refluxing method used in the widely used Berger-Collela timestepping strategy [27].

Overall, AMR-based methods can produce significant speedup for applications dominated by flow features highly localized in space, which is especially appealing for steady hydrodynamic phenomena (e.g. external shock-containing flow) and canonical unsteady shock-dominated flows. In these scenarios, the added costs associated with memory management stemming from fine-grid initialization is insignificant compared to the reduction in the number of degrees of freedom [367]. The primary issue with AMR, however, is that if the desired flowfields are unsteady, grid movement at finer levels can produce significant errors in characterizing turbulent combustion processes that are non-local in nature [314], such as fuel stratification effects observed in RDEs [272]. Additionally, in applications where features of interest span the entire domain (e.g. domain-wide pressure waves that drive thermoacoustic instabilities in gas turbine combustors or turbulent eddies in a channel flow), AMR loses acceleration capability due to the fact that the generated fine grids must also span the entire domain.

2.4.4 Approximate Inertial Manifolds (AIM)

Approximate inertial manifolds (AIM), popularized in Ref. [92, 334], are a physics-based method for producing ROMs for fluid flows governed by parabolic PDEs, which upon spatial discretization (Eq. 2.12) reduce to nonlinear dynamical systems heavily influenced by diffusion. As such, the method has been successfully used to produce physics-based ROMs for canonical turbulent flows and turbulent reacting flows [5, 6], resulting in interpretations of the AIM approach as a type of LES subgrid scale model [4] or nonlinear Galerkin method [193].

In the AIM modeling approach, the diffusive, linear, and self-adjoint operator $\mathcal{D}$ in Eq. 2.12 is assumed to invoke a manifold $\mathcal{M}$, which exists in a subset of the full-dimensional phase space. This manifold, among other properties, exponentially attracts all solutions of Eq. 2.12; in other words, all initial conditions in the N_F dimensional phase space will eventually lie within the manifold $\mathcal{M}$ in the long-time limit.

In AIM, orthogonal projection operators that rotate the state vector $\mathbf{y}$ into an alternate, manifold-conforming space are derived from an eigendecomposition of the linear diffusive operator $\mathcal{D}$. Because the diffusive operator in the compressible Navier-Stokes equation is a Laplacian, the decomposition results in an orthonormal basis that can be used to split the state-space into resolved and unresolved components, similar to the concept of filtering in LES [4]. Because the resolved and unresolved components in the projected space are uncorrelated, they can be treated as independently evolving dynamical systems. The objective of AIM is to (a) retain only the resolved components in the simulation, and (b) account for the effect of the unresolved components on the resolved components using manifold properties.

Using the full-order state vector $\mathbf{y} \in \mathbb{R}^{N_F}$, the resolved and unresolved projections are given as $\mathbf{w} = P\mathbf{y}$ and $\mathbf{u} = Q\mathbf{y}$ respectively. The projection operators P and Q are derived from subsets of the eigenvectors of $\mathcal{A}$ based on the eigenvalue distribution. Known as the spectral gap condition [334], the idea is to choose P such that its corresponding smallest eigenvalue is much larger than the largest eigenvalue of Q. If the eigenvalues of $\mathcal{A}$ are sorted in ascending order as

$$0 < \lambda_1 < \lambda_2 < \ldots < \lambda_{N_F}, \tag{2.22}$$

one can interpret the spectral gap condition as the index i at which the ratio λ_{i+1}/λ_i is maximized. The dimensionality of the resolved modes in $\mathbf{u}$, which provides the degree of order reduction, can then be extracted from the cutoff index i.

The evolution equations for the resolved and unresolved scales are, respectively,

$$\frac{d\mathbf{w}}{dt} = \mathcal{D}\mathbf{w} + P\mathcal{A}(\mathbf{y}) + P\mathcal{R}(\mathbf{y}), \quad \mathbf{w}(t=0) = P\mathbf{y}_0 \tag{2.23}$$

and

$$\frac{d\mathbf{u}}{dt} = \mathcal{D}\mathbf{u} + Q\mathcal{A}(\mathbf{y}) + Q\mathcal{R}(\mathbf{y}), \quad \mathbf{u}(t=0) = Q\mathbf{y}_0. \tag{2.24}$$

In AIM, only Eq. 2.23 is treated numerically – analogous to LES, the modeling task comes from the closure problem of estimating the full state vector $\mathbf{y}$ from only the resolved components $\mathbf{w}$ such that the nonlinear terms can be evaluated. This closure problem is addressed by invoking a manifold assumption through a so-called slaving principle. The argument is that an existence of a manifold implies that the unresolved scales instantaneously relax to the conditions prescribed by the resolved scales, such that their rates can be set to zero:

$$\frac{d\mathbf{u}}{dt} = \mathcal{D}\mathbf{u} + Q\mathcal{A}(\mathbf{y}) + Q\mathcal{R}(\mathbf{y}) = 0. \tag{2.25}$$

By expressing the full-order state vector as $\mathbf{y} = P\mathbf{w} + Q\mathbf{u}$, the convergence of implicit Newton iterations on Eq. 2.25 based on an initial guess for $\mathbf{w}$ produces an approximation to the full flowfield $\mathbf{y}$, which can then be used to close the nonlinear terms in Eq. 2.23. Figure. 2.9 shows an application of the AIM procedure on modeling the resolved scales in homogeneous isotropic turbulence. Although the method can produce significant reduction in computational time in canonical turbulent flows combustion phenomena such as extinction/re-ignition [4], extensions of the AIM approach to flows dominated by advection and reaction are invalid due to the inherent assumption that a large part of the flow energy must come from the diffusion operator (other methods such as computational singular perturbation, described in Sec 2.4.5, can address this limitation). Additionally, unless the diffusion operator is already diagonalized (this can only be true in periodic domains), another issue is that the projection operators come from a potentially prohibitive eigendecomposition of a large Laplacian matrix.

2.4.5 Computational Singular Perturbation (CSP)

CSP is another physics-based modeling approach based on manifold theory that is similar conceptually to AIM, but focuses primarily on the chemical reaction contribution (the chemical source term) instead of the turbulence contribution (diffusion). Since the introduction of the technique by *Lam and Goussis* [164], CSP has been successfully used to accelerate stiff chemistry time integration routines that arise in the simulation of compressible reacting flow using detailed chemistry [111, 365].

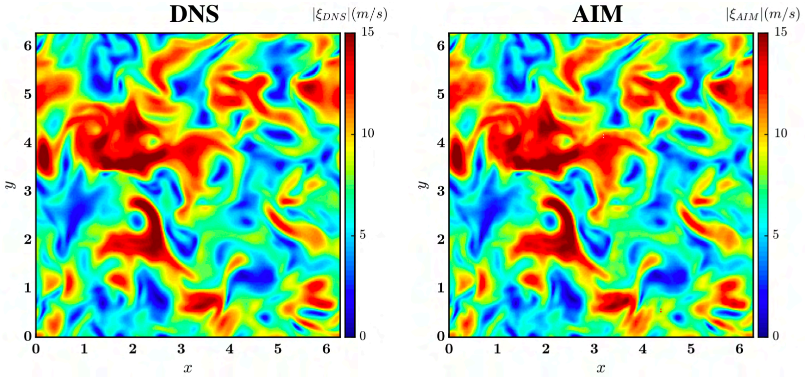


Figure 2.9: **(Left)** Velocity field magnitude for resolved scales from direct numerical simulations of homogeneous isotropic turbulence. **(Right)** Modeled resolved scales using AIM approach. Reproduced from Ref. [6].

To describe CSP in clear terms, instead of treating the original system in Eq. 2.12, we treat the nonlinear ODE that governs the evolution of the thermochemical state vector $\phi \in \mathbb{R}^{N_S+1}$,

$$\frac{d\phi}{dt} = S(\phi), \tag{2.26}$$

where ϕ consists of species concentrations and some variable for energy (usually temperature), and S is the chemical source term. Equation 2.26 is a highly stiff nonlinear dynamical system used to model autoignition processes at either constant volume or constant pressure[1]. Similar to AIM, the idea is to use a basis projection to separate the full state vector into resolved modes and unresolved modes. Instead of deriving the basis from the decomposition of the linear operator $\mathcal{A}$, CSP derives the basis from the Jacobian of the chemical source term,

$$\mathbf{J} = \frac{\partial S(\phi)}{\partial \phi}. \tag{2.27}$$

The goal in CSP is to eliminate chemical stiffness through these basis projections. The resolved modes, which are solved for numerically, contain only the slow chemical timescales, whereas the unresolved modes containing the fast chemical timescales are

[1] The reader is pointed to Chapter 1 in the textbook by *Peters* [252] for a derivation of the ODE in Eq. 2.26 for constant-pressure reactors.

modeled. It can be shown that the eigenvectors of the chemical Jacobian provide a leading-order approximation of the true manifold, termed by Maas and Pope as the intrinsic low-dimensional manifold [190, 341]. Since the fundamental modeling procedure is the same as in AIM, equations used in CSP will not be repeated here: the projection operators P and Q are extracted from the eigenvectors of the chemical Jacobian based on a spectral gap condition, and the closure model for the impact of the unresolved (fast) scales on the dynamics of the resolved (slow) scales is derived algebraically via a slaving principle through Newton iterations on Eq. 2.25.

As implied above, the primary difference between CSP and AIM is in the derivation of the basis functions. In CSP, the basis functions are derived from the eigendecomposition of the Jacobian of a nonlinear reaction source term operator, which means the projection operators are functions of the state, and are therefore time-evolving. This also implicitly assumes that the energy budget of the flowfield is dominated by chemical reactions as opposed to diffusion terms as in AIM. Additionally, because the basis functions must be updated as the state variables change during time integration, significant computational cost is added to the CSP modeling approach [101]. The hope is that the timestep savings provided by eliminating the fast scales offsets the additional cost incurred by eigendecompositions, which scale in complexity as as the cube of the state vector dimensionality and linearly with the number of cells in the computational domain. Lastly, it should be noted that the idea of CSP – namely, using an eigendecomposition of the RHS Jacobian to obtain a time-varying basis – can be applied to any arbitrary dynamical system RHS, such as the full high-fidelity system in Eq. 2.12. This is typically not done, however, because computing full-system Jacobians is infeasible in practical applications. For this reason, applications of CSP are almost exclusively used for thermochemical state transport based on the chemical source term contribution in isolation (Eq. 2.26) within operator splitting frameworks described in the next section.

2.4.6 Operator Splitting

Although not technically a modeling approach, operator splitting is included here in the class of phyics-based modeling strategies because the method induces a residual on the underlying dynamical system of Eq. 2.12. As discussed below, this residual itself is not addressed or accounted for in the method (which means it is not technically a model), but since it provides simulation acceleration in a practical way at the cost of introducing error into the equations, it is interpreted here as a modeling strategy.

Initially presented and popularized by *Strang* [325], the class of operator splitting methods are models that decouple the physical processes in the underlying dynamical system of Eq. 2.12. Interpretations of operator splitting as a type of physics-based modeling framework have been made through the lens of manifold methods like CSP [321].

Operator splitting methods are ideal when a single operator – for example, the chemical source term contribution – is responsible for the system stiffness, and the timescales required to resolve the dynamics stemming from the remaining operators (e.g. acoustic or diffusive timescales) are not prohibitive [166]. As decribed in Sec. 1.4, this is the scenario in the simulation of advection-dominated compressible reacting flow such as detonations. The central idea in operator splitting is to decouple the physical processes (i.e. $\mathcal{A}$, $\mathcal{D}$, and $\mathcal{R}$ in Eq. 2.12) during a single simulation timestep prescribed by the CFL condition, allowing the user to target optimal time integration routines for each physical phenomena in isolation. This concept, shown in Fig. 2.10 for the case of symmetric Strang splitting [325], is particularly useful in chemically reacting flows because (1) stiff solvers or models for chemistry (e.g. CSP-based methods) can be linked to the flow solver as external modules to essentially eliminate the originally prohibitive cost associated with chemical stiffness, and (2) stability-preserving explicit methods based on the more reasonable CFL timescale, combined with well established approximate Riemann solvers and turbulence models, can be

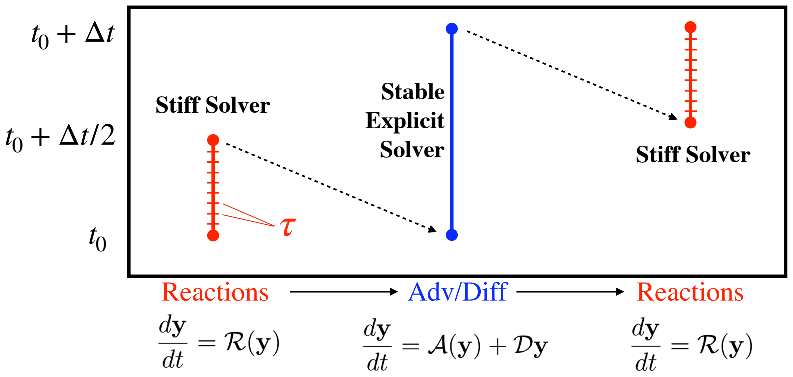

$t_0 + \Delta t$
$t_0 + \Delta t/2$
t_0
Stiff Solver
Stable Explicit Solver
Stiff Solver
τ
Reactions
Adv/Diff
Reactions
$\frac{d\mathbf{y}}{dt} = \mathcal{R}(\mathbf{y})$
$\frac{d\mathbf{y}}{dt} = \mathcal{A}(\mathbf{y}) + \mathcal{D}\mathbf{y}$
$\frac{d\mathbf{y}}{dt} = \mathcal{R}(\mathbf{y})$

Figure 2.10: Diagram of Strang splitting approach used to solve Eq. 2.12, where t_0 denotes the initial time, Δt the simulation timestep, and τ the chemical timescale. Dashed arrows indicate state initialization.

used to treat advection and diffusion terms. In this sense, operator splitting methods do not achieve acceleration by reducing system dimensionality; instead, computational cost is alleviated through the decoupling procedure by allowing the user to deploy targeted algorithms for each respective physical source in isolation.

The widespread usage of operator splitting is due in part to the 2nd order temporal accuracy provided by one such variant known as Strang splitting [321]. However, the decoupling procedure introduces a residual, called the *splitting error*, that causes the operator-split system to deviate from the baseline fully-coupled system – since this deviation is a forcing term that is not accounted for, it resembles a truncation or model-form error [166, 325]. More formally, if the flowfield obtained using operator splitting is denoted $\tilde{\mathbf{y}}$, then its governing equation can be described in a similar way as other modeling approaches like LES (Eq. 2.15) and AMR (Eq. 2.19) as

$$\frac{d\tilde{\mathbf{y}}}{dt} = \mathbf{F}(\tilde{\mathbf{y}}) + \mathbf{M}(\tilde{\mathbf{y}}). \tag{2.28}$$

For Strang splitting, the residual $\mathbf{M}$ is proportional to (1) the square of the simulation timestep, Δt, and (2) the limiting timescales of each of the operators in Eq. 2.12 – as shown in Ref. [166], the residual can be analytically expressed using the operator Jacobians as $\mathbf{M} = \Delta t^2 f(\frac{\partial \mathcal{D}\tilde{\mathbf{y}}}{\partial \tilde{\mathbf{y}}}, \frac{\partial \mathcal{A}(\tilde{\mathbf{y}})}{\partial \tilde{\mathbf{y}}}, \frac{\partial \mathcal{R}(\tilde{\mathbf{y}})}{\partial \tilde{\mathbf{y}}})$, where f is a linear function. Although operator splitting is widely used due to its compatibility with other combustion modeling frameworks [268], splitting errors are known to pollute reacting flow simulations by misrerpresenting key combustion processes and flow features such as autoignition [357, 362], extinction [184], and detonation wave structures [17]. Additionally, when the ratio between the simulation timestep and the limiting timescale is extremely high, convergence trends may deviate from theoretical second-order expectations [321]. Lastly, operator splitting should not be used when (1) one requires a global temporal discretization error to be higher second order, and (2) when the limiting timescales of all physical processes at play (e.g. advection, diffusion, reaction) are all of the same order. However, because the decoupling procedure allowed by operator splitting leads to (a) highly stable and efficient (albeit low-order) numerical schemes, and (b) compatibility with optimized libraries for both kinetics evaluations [16] and stiff time integration [63, 111], this method continues to be widely used in almost all numerical combustion applications at varying levels of fidelity.

2.4.7 Summary and Limitations of Physics-Based Models

A summary of the physics-based approaches discussed above is presented in Table 2.1. The above discussion outlined a few key pathways for physics-based modeling that are routinely used to accelerate high-fidelity simulations of reacting flow. Overall, the primary advantage in these models is interpretability – model-form errors can easily be accounted for, as the starting point in all of the above approaches is the governing equations (Eq. 2.11) and any present parameters are tuned by field experts. The disadvantage, however, is lack model extendability (also referred to as non-universality [276]). Physics-based models by design are derived from simple canonical configurations, and therefore cannot be extended reliably to complex, full-geometry simulations that contain a wide variety of flow turbulent combustion

Modeling Approach	**Modeling Goal / Residual**	**Acceleration Mode**	**Target Application**	**Key References**
Large-eddy Simulation (Sec. 2.4.1)	Eq. 2.16	Fixed order reduction	Canonical turbulence	[165, 258, 265]
Flamelet / Progress Variable (Sec. 2.4.2)	Eq. 2.18	Chemistry tabulation	Low-Mach turbulent combustion	[252, 256, 258]
Adaptive Mesh Refinement (Sec. 2.4.3)	Eq. 2.21	Dynamic order reduction	Advection-dominated flows	[27]
Approximate Inertial Manifolds (Sec. 2.4.4)	Eq. 2.25	Fixed order reduction	Diffusion-dominated flows	[4, 334]
Computational Singular Perturbation (Sec. 2.4.5)	Same as AIM	Stiffness elimination	Stiff reacting flows	[111, 164]
Operator Splitting (Sec. 2.4.6)	Eq. 2.28	Decoupling forcing functions	General multi-physics flows	[166, 325]

Table 2.1: Summary of physics-based models used to accelerate high-fidelity simulations of the Navier-Stokes equations.

regimes that may or may not be dominated by diffusion-based (where LES excels), advection-based (where AMR excels), or reaction-based processes (where CSP and flamelet models excel). In light of the shortcomings of physics-based models, Sec. 2.5 describes a different strategy, namely data-based modeling, that alleviates some of these issues.

2.5 Data-Based Models

Data-based modeling has emerged as a useful pathway for simulation acceleration in recent years [79, 156]. The popularity of these methods comes not only from the unprecedented availability of high-quality data in the fluid dynamics community both from numerical simulations and experimental diagnostics [276], but also from the advances in HPC-compatible algorithms developed in the data science, artificial intellgience (AI), and machine learning (ML) communities [41]. Within the field of numerical combustion, this has led to the fusion and replacement of many conventional modeling approaches with ML strategies [78, 369]. The rate of adoption of ML/AI approaches that thrive in data-rich environments for simulation acceleration purposes is exacerbated by the now widespread availability of high-fidelity reacting flow solver suites capable of leveraging the exascale-ready class of supercomputers [7, 30, 55, 349]. Ultimately, as data continues to grow, modeling approaches that scale with available data are required.

Instead of starting from the governing equations directly, data-based models achieve simulation acceleration by constructing models that optimize global objective functions based on *training data*, which are high-quality datasets derived from the governing equations in some way. For example, the training datasets can be obtained from the governing equations directly using a small number of data-rich high-fidelity DNS trajecories (e.g. from solutions to Eq. 2.12), or indirectly, via observable functions of the underlying flowfields through experimental inquiry (e.g. planar flowfield measurements and laser diagnostics). Data-based model development is comprised of two stages: (1) an offline training stage, where algorithms are used to tune model parameters such that objective functions are optimized, and (2) the deployment or inference stage, where the trained models are actually used to accelerate flow solvers. Most of the computational effort in data-based modeling comes from the training stage – the hope is that this high offline cost is offset by achieving significant acceleration in the deployment stage.

A survey of several popular data-based techniques used to achieve simulation acceleration within the CFD and numerical combustion communities is provided in the sections below. These methods can be classified into either the *unsupervised* or *supervised* modeling categories [214]. Unsupervised data-based models cast objective functions as functions of only the training dataset – their goal is to produce models based on flowfield compression by extracting only the salient features of the original data, where the quantitative measure for salience is defined implicitly through the objective function. On the other hand, supervised models cast objectives functions as functions of two datasets, the training dataset and a target dataset, where the target dataset contains samples of the modeling quantity of interest that is assumed to be statistically correlated to the training dataset samples. The goal in supervised methods is to parameterize this correlation with efficient mapping functions (e.g. neural networks). As with the physics-based models in Sec. 2.4, this overview is not meant to cover every single data-based method used in the fluid dynamics community (there are hundreds if not thousands of variations), but rather a few key frameworks that see widespread use for the purpose of accelerating high-fidelity reacting flow solvers.

2.5.1 Modal Decomposition via Space-Time Decoupling

Modal decomposition methods can be used to achieve order reduction, and therefore simulation acceleration, by decoupling the space and time components of the flowfield $\mathbf{y}(t)$ (Eq. 2.12) [77, 330]. The decomposition is described in a flowfield reconstruction procedure using a linear combination of a finite set of spatial modes and temporal coefficients as

$$\widehat{\mathbf{y}}(t) = \sum_{m=1}^{M} \mathbf{a}_m(t)\mathbf{u}_m, \tag{2.29}$$

where M denotes the total number of retained spatial modes in the decomposition. In Eq. 2.29, $\mathbf{u}_m$ is the m-th spatial mode that is fixed in time – these spatial modes exist in the same phase space as the flowfield, and are therefore interpreted as visualizable flow directions that can be analyzed like any other flow variable. The quantity $\mathbf{a}_m(t)$ indexes the m-th component of the vector $\mathbf{a} \in \mathbb{R}^M$, which contains the full set of M time-evolving coefficients, called the temporal coefficents, that encode the contribution/importance of each of the respective modes on the flowfield at a single time instant t. It is emphasized that the formulation in Eq. 2.29 is a single approach to the general class of modal decomposition methods – modal decompositions do not necessarily have to take the form of fixed-in-time spatial modes with temporal coefficients encoding time variation. For example, methods that extend the above decomposition with time-evolving spatial modes (i.e. $\mathbf{u}_m = \mathbf{u}_m(t)$) have been explored to good effect [83, 279]. In the survey described hereafter, the context of modal decomposition applies to static spatial modes and time-evolving temporal coefficients, as many of the techniques used to accelerate reacting flow simulations (discussed next in Sec. 2.5.2-2.5.4) utilize this framework.

The error incurred in the decomposition (i.e. deviation $\widehat{\mathbf{y}}(t)$ from the true flowfield $\mathbf{y}(t)$) depends on both the number of modes M and the method by which the modes are recovered from the underlying dataset. This dataset is given as a matrix $\mathcal{Y} =$

$[\mathbf{y}(t_1), \mathbf{y}(t_2), \ldots, \mathbf{y}(t_N)] \in \mathbb{R}^{N_F \times N}$; its columns contain a set of N_F-dimensional high-fidelity flowfield samples (referred to as snapshots), and its rows represent the N time samples of the spatial discretization points in each of the snapshots. The data matrix $\mathcal{Y}$ can be populated by means of high-fidelity solutions of the dynamical system in Eq. 2.12 or by time-resolved experimental diagonstic measurements.

Modal decomposition methods based on space-time decoupling are used in two contexts: (1) conducting an expert-guided analysis of the spatial modes, which may contain physically relevant information of the system under study , and (2) constructing reduced-order models (ROMs) that describe the evolution of the temporal coefficients. The modal decomposition based ROM must ensure that the transport model for $\mathbf{a}(t)$ results in a faster alternative to solving the high-fidelity solutions of the full-order dynamical system that describes the evolution of $\mathbf{y}$ – in other words, M (the number of retained modes) must be significantly smaller than N_F (the flowfield dimensionality).

The various modal decomposition strategies outlined below – namely proper orthogonal decomposition (Sec. 2.5.2) [29], dynamic mode decomposition (Sec. 2.5.3) [300], and cluster-based reduced order modeling (Sec. 2.5.4) [140] – are similar in that they obey the same decomposition goal described in Eq. 2.29. The difference between these methods, however, is in how they go about extracting the $\mathbf{u}_i$ spatial modes from the dataset $\mathcal{Y}$ using optimization strategies, and how order reduction facilitated by projection onto these modes is used to accelerate simulations. Nonlinear extensions of decomposition strategies based on Eq. 2.29 are provided by autoencoders, described in Sec. 2.5.5.

There are many ways to derive methods for transporting the temporal coefficients $\mathbf{a}(\mathbf{t})$ using the spatial modes [330]. In one strategy, the transport rule for the temporal coefficients can be extracted by projecting the spatial modes onto the governing PDEs using linear or non-linear Galerkin-type approaches [161, 169, 316]. In an alter-

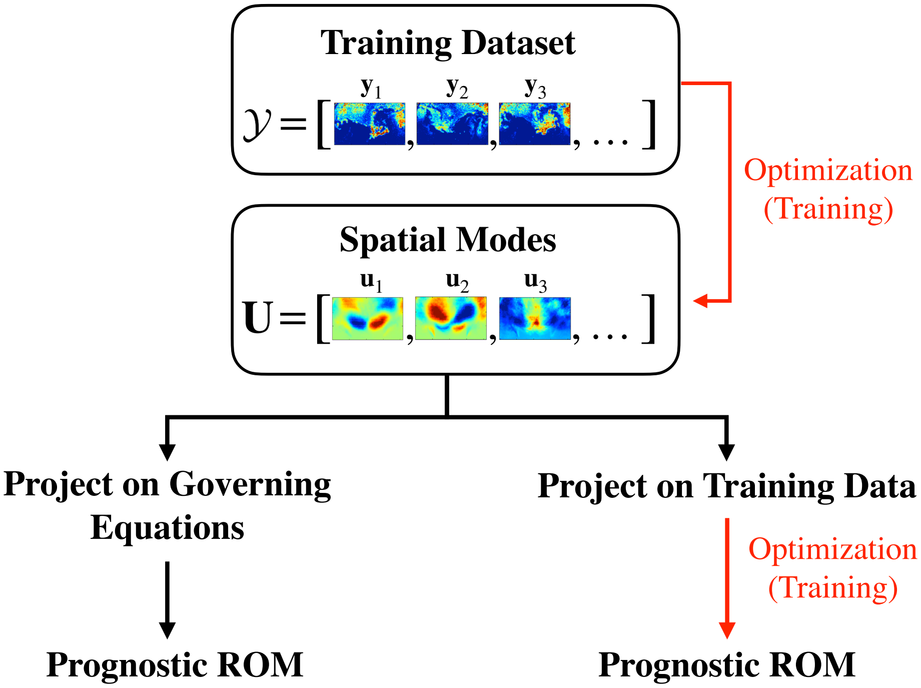

Training Dataset
$\mathcal{Y}=[\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \ldots]$
Optimization
(Training)
Spatial Modes
$\mathbf{U}=[\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \ldots]$
Project on Governing Equations
Project on Training Data
Optimization
(Training)
Prognostic ROM
Prognostic ROM

Figure 2.11: Pathways for data-based reduced order model development within the modal decomposition framework based on space-time decoupling.

nate strategy, temporal coefficient dynamics are evaluated a-priori by projecting the snapshot data $\mathcal{Y}$ onto the reatined spatial modes, producing a "reduced" dataset $\mathcal{A}$. Then, separate data-based optimization strategies can be used to produce lightweight models for predicting the temporal coefficient dynamics in $\mathcal{A}$ instead of $\mathcal{Y}$ (e.g. supervised training of recurrent neural networks [199], residual networks [117], dynamic mode decomposition [300], and other data-based strategies for predicting time series [198]) . This procedure is outlined in Fig. 2.11.

2.5.2 Proper Orthogonal Decomposition (POD)

POD is one of the most widely used modal decomposition techniques because it provides optimal linear compression; it directly optimizes the flowfield reconstruction in Eq. 2.29 using the lowest number of spatial modes. Popularized in *Sirovich* [315] and *Berkooz et al.* [29] for the description of turbulent flows, POD is formally equivalent to principal component analysis [356] and Karhunen-Loeve decomposition

[40]. This unsupervised decomposition procedure outputs the set of spatial modes – here represented as the matrix $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_M]$ – from an eigendecomposition of the data covariance matrix. The covariance matrix can be derived from the centered data matrix $\widetilde{\mathcal{Y}}$ as

$$\mathbf{C} = \frac{1}{N}\left(\widetilde{\mathcal{Y}}\widetilde{\mathcal{Y}}^{\mathrm{T}}\right) = \mathbf{V}\Lambda\mathbf{V}^{-1} \in \mathbb{R}^{N_F \times N_F}, \tag{2.30}$$

where the superscript T denotes a matrix transpose and the set of $\mathbf{u}_m$ modes are recovered from the eigenvectors of the covariance matrix $\mathbf{V}$. Since $\mathbf{C}$ is symmetric and positive semidefinite, the $\mathbf{u}_m$ modes contained in $\mathbf{V}$ are real and orthonormal, i.e. $\mathbf{V}^{-1} = \mathbf{V}^{\mathrm{T}}$ and $\langle \mathbf{u}_i, \mathbf{u}_j \rangle = \delta_{ij}$.

The total number of available modes is M_{full}, and is bounded by the rank of the data matrix $\mathcal{Y}$. The number of retained modes, on the other hand, is $M \leq M_{full}$. The retained modes, referred to as the POD modes, are collected in the columns of the matrix $\mathbf{U}$ and can be interpreted as a subset of the eigenvector matrix $\mathbf{V}$. Projection of the data matrix $\mathcal{Y}$ onto the subspace spanned by the M POD modes produces the $M \times N$ temporal coefficient matrix $\mathcal{A}$ as

$$\mathcal{A} = \mathbf{U}^{\mathrm{T}}\mathcal{Y}, \tag{2.31}$$

where the instantaneous coefficient vectors $\mathbf{a}(t)$ are contained in the columns of $\mathcal{A}$, and the m-th row of $\mathcal{A}$ describes the time evolution of a scalar coefficient for the m-th mode.

Since the POD modes diagonalize the covariance matrix of the data [315], the eigenvalue Λ_m represents the energy contribution to the flow from mode $\mathbf{u}_m$. The mode corresponding to the highest eigenvalue of $\mathbf{C}$ thus captures the most amount of flow "energy" or variance. In POD-based analysis, assuming the modes are arranged by their eigenvalues in descending order, this attribute guides the question of how to determined M: the first M modes are kept such that a sufficiently high amount of the flow energy is resolved (typically over 90%) [29]. POD is most advantageous when a small number of modes contains a large amount of the flow energy, which amounts to a spectral gap condition in the spectrum of the covariance $\mathbf{C}$ (this is analogous to physics-based methods like CSP and AIM). The advantage of POD is that energy captured by the first M modes is easily accessed by the sum of the first M eigenvalues of the covariance matrix. If these modes are contained in the first M columns of the matrix $U \in \mathbb{R}^{N_F \times M}$, it can be shown that the POD formulation in Eq. 2.30 solves the following optimization problem:

$$\min_{U} \|\mathcal{Y} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathcal{Y}\|_F, \quad \text{s.t.} \quad \mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}, \tag{2.32}$$

where F is the Frobenius norm and $\mathbf{I}$ is the size N_F identity matrix . In other words, POD provides an orthogonal set of modes in $\mathbf{U}$ that optimally reconstruct the snapshot set $\mathcal{Y}$. The reconstruction error vanishes when $M = M_{full}$.

Overall, POD has facilitated better understanding of complex flow phenomena and has led to a wide variety of data-driven ROM frameworks for complex compressible reacting flow applications, from low-Mach turbulent combustion [91, 134, 229, 322] to advection-dominated combustion settings found in scramjets and rotating detonation engines [303, 363, 368]. It should be noted that the utilization of POD in many of these complex full-geometry applications is restricted exclusively to modal analysis (i.e. visual inspection of the POD modes obtained from DNS or experimental flowfields) – development of robust predictive frameworks for the evolution of temporal coefficients is, with some exceptions, primarily restricted to canonical flows [127, 179, 198, 293]. Additionally, POD faces disadvantages in capturing extreme events that may stem from low-energy contributions, which are by design not retained in the mode truncation [122]. Despite these shortcomings, due to its simplicity and optimal nature, POD continues to see widespread usage and has led to the development of several variant such as gappy POD [353], balanced POD [286], and spectral POD [337]) among others, each with their own targeted applications.

2.5.3 Dynamic Mode Decomposition (DMD)

Introduced in *Schmid* [300] and inspired by the initial work in *Mezić* [205], DMD has gained significant popularity in the past decade as a data-based reduced order modeling method and shares significant overlap with discrete Fourier decompositions and spectral analysis [287, 337]. Although it is a modal decomposition technique, DMD is fundamentally different from POD as it models the time evolution of the state vector $\mathbf{y}$ using linear dynamics.

More specifically, we define the discrete-time dynamical system that acts on the snapshots in $\mathcal{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N]$ as the nonlinear propagator $\mathcal{F}$, such that

$$\mathbf{y_{s+1}} = \boldsymbol{\mathcal{F}}(\mathbf{y_s}). \tag{2.33}$$

The operator $\mathcal{F}$ can be derived from the underlying nonlinear dynamical system in Eq. 2.23. As described in Ref. [339], DMD approximates the above equation using linear dynamics via

$$\mathbf{y}_{s+1} = \mathbf{A}_{DMD}\mathbf{y}_s. \tag{2.34}$$

In Eq. 2.34, $\mathbf{A}_{DMD}$ is stationary linear operator derived from a least-squares regression approach using the data matrix $\mathcal{Y}$. The linear model in Eq. 2.34 has close connections with Koopman operator theory [206] – a brief overview of these connections are provided in the following paragraphs, as is the procedure by which one obtains the linear operator from $\mathcal{Y}$ and the associated DMD modes.

Consider an arbitrarily defined observable function (which can be scalar or vector-valued) of the state, $g(\mathbf{y})$. The Koopman operator $\mathcal{U}$ of the dynamical system is the linear propagator of any such observable,

$$\mathcal{U}g(\mathbf{y}_s) = g(\boldsymbol{\mathcal{F}}(\mathbf{y})). \tag{2.35}$$

If an observable function ϕ_j satisfies the time evolution given by

$$\mathcal{U}\phi_j(\mathbf{y}_s) = \lambda_j \phi_j(\mathbf{y}_s), \tag{2.36}$$

then the function ϕ_j is called a Koopman eigenfunction. A primary assumption in DMD is that the Koopman operator can be described purely by a discrete spectrum [205]. That is, any observable is contained within the span of the eigenfunctions of the Koopman operator and can be expressed by the expansion

$$g(\mathbf{y}_s) = \sum_{j=1}^{\infty} \phi_j(\mathbf{y}_s)\mathbf{v}_j, \tag{2.37}$$

where $\mathbf{v}_j$ is called the Koopman mode. If such an expansion is valid, then the evolution of the observable can be expressed directly in terms of the Koopman modes and eigenvalues as

$$\mathcal{U}^t g(\mathbf{y}_s) = g(\mathbf{y}_{s+t}) = \sum_{j=1}^{\infty} \lambda_j^t \mathbf{v}_j, \tag{2.38}$$

where the initial condition information $\phi_j(\mathbf{y}_s)$ has been absorbed into the Koopman mode. The discrete spectrum assumption is valid if the dynamics described by $\boldsymbol{\mathcal{F}}$ is a truly periodic or linear system. To extend Koopman analysis to nonlinear systems that are quasi-periodic or chaotic, there is also a continuous portion of the $\mathcal{U}$ spectrum that is orthogonal to the space spanned by the eigenfunctions that should be accounted for in the expansion of $g(\mathbf{x}_s)$. For more on this topic, see Refs. [205] and [206].

Using the identity observable function $g(\mathbf{y}_s) = \mathbf{y}_s$, one can connect the Koopman expansion above to the dynamics of the system $\boldsymbol{\mathcal{F}}$. It can be shown that if $\boldsymbol{\mathcal{F}}$ is linear, and the dynamics is described by a linear propagator $\mathbf{A} : \mathbb{R}^{N_p} \mapsto \mathbb{R}^{N_p}$ where $\mathbf{y}_{s+1} = A\mathbf{y}_s$, then the operator A is the Koopman operator and the eigenvectors of A are the Koopman modes $\mathbf{v}_j$ [206, 287].

The underlying concept of DMD comes from this connection between the Koopman operator and the linear operator $\mathbf{A}$. Since the actual system $\boldsymbol{\mathcal{F}}$ is in fact nonlinear (Eq. 2.12), no true $\mathbf{A}$ exists for the system. In standard DMD algorithms, which deal with the identity observable function, a linear operator $\mathbf{A}_{DMD}$ of Eq. 2.34 is instead approximated from the nonlinear data using regression techniques. Once such an $\mathbf{A}_{DMD}$ is recovered, its eigenvalues and eigenvectors provide the DMD temporal coefficients and spatial modes used in the modal decomposition of Eq. 2.29. These quantities approximate the Koopman eigenvalues and modes from Eq. 2.38.

If the data in $\mathcal{Y}$ is arranged in snapshot pairs such that $\mathcal{Y}_1 = [\mathbf{y}_1, \dots, \mathbf{y}_{N-1}]$ and $\mathcal{Y}_2 = [\mathbf{y}_2, \dots, \mathbf{y}_N]$ (*i. e.* $\mathcal{Y}_2 = \boldsymbol{\mathcal{F}}(\mathcal{Y}_1)$), then $\mathbf{A}_{DMD}$ can be recovered by finding a linear map between $\mathcal{Y}_1$ and $\mathcal{Y}_2$ in the least squares sense. This is accomplished with the pseudo-inverse of $\mathcal{Y}_1$: $\mathbf{A}_{DMD} = \mathcal{Y}_1^+ \mathcal{Y}_2$, where + denotes the pseudo-inverse. As implied by Eq. 2.38, the complex eigenvectors of $\mathbf{A}_{DMD}$ yield the full set of DMD modes; the complex eigenvalues provide the temporal coefficients. The real part of the eigenvalues provides a decay rate of the dynamics projected onto the respective modes, and the imaginary part provides oscillation frequencies which are fixed for each coefficient. This is very similar to Fourier-type decompositions – in fact, it can be shown that DMD produces the same results as discrete Fourier transforms under certain assumptions [56].

As with POD, DMD has been used for both modal analysis/visualization as well as predictive modeling [339]. Specific applications include predicting canonical turbulent flows [102], analysis of swirl flames [194], analysis of thermoacoustic instabilities in gas turbine combustors [211], and modeling wake dynamics in scramjet engines

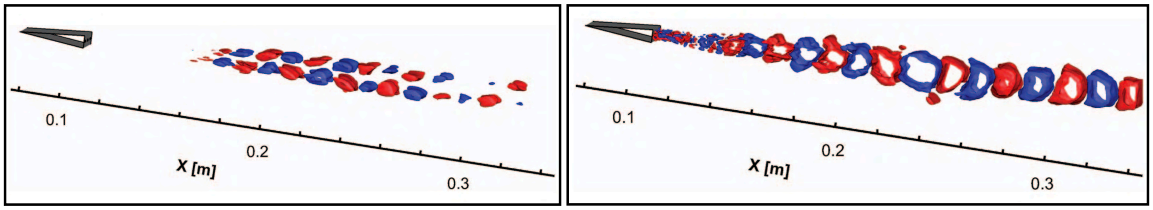


Figure 2.12: Visualization of two DMD modes extracted from large-eddy simulations of a model scramjet combustor (reproduced from Ref. [177]).

[177] (see Fig. 2.12). Due to the limitations of constant-frequency temporal coefficients, standard DMD algorithms often struggles to predict multi-frequency transition phenomena and highly transient flow that are often present in turbulent combustion applications. Additionally, for physical interpretation and analysis, it is not realistic to retain all DMD modes – as with POD, a truncation from the full set of M_{full} modes to a reduced set of size M is required to achieve order reduction. However, mode selection in DMD is not as clear-cut as with POD [339] – common metrics are to use either 1) growth rate (keep the modes which decay the least over successive operations of $\mathbf{A}_{DMD}$), or 2) mode amplitude (selecting the modes with the highest L_2 norm). Exclusively using mode amplitude as the criteria for mode selection might result in choosing a mode active for a very short amount of time. Similarly, selecting the mode based on growth factor alone may lead to a mode that is slowly decaying but of minimal amplitude. These ambiguities often force the end-user to carry out expert-guided visual analysis of the flow patterns contained within the modes, which is a time-consuming post-processing step.

The class of DMD-based methods continues to evolve to address these limitations and is an active field of research [301]. Powerful extensions to the baseline DMD formulation provided above involve kernel strategies where DMD is applied in a transformed latent space that better satisfies the linear dynamics constraint [187, 239, 240]. Movement into a latent space is often facilitated by an autoencoder, as described in Sec. 2.5.5.

2.5.4 Cluster-based reduced-order modeling (CROM)

Cluster-based reduced order modeling (CROM) is another data-based modeling approached connected to modal decompositions introduced in the work of *Kaiser et al.* [140]. Instead of modeling dynamics of individual phase space trajectories, CROM constructs a reduced-order model for the underlying Liouville equations which transport PDFs of the state [103]. The crux of the approach is to use an unsupervised K-means clustering algorithm on the snapshot data $\mathcal{Y}$ to discretize the phase space into a set of non-overlapping clusters. By minimizing a cluster-based optimization problem, the output clusters are conditioned to represent only the regions explored by the trajectories contained in $\mathcal{Y}$. More specifically, the clustering via K-means produces a data-adapted Voronoi tesselation (or mesh) of the N_F-dimensional phase space (shown in Fig. 2.13. The spatial modes in CROM are referred to as centroids, which are computed as regional averages of all the snapshots in the corresponding clusters. The temporal coefficients are obtained through a one-hot encoding, or classification, that informs by means of Euclidean distances the cluster in which a given instantaneous snapshot $\mathbf{y}$ resides.

The primary output of CROM is a prognostic model, called the transition matrix, that estimates the probability of snapshot transitions between clusters as

$$\mathbf{p}_{t+\Delta t} = \mathcal{P}\mathbf{p}_t \in \mathbb{R}^M, \tag{2.39}$$

where $\mathbf{p}$ encode the cluster probabilities, and $\mathcal{P}$ is a Markovian transition matrix extracted from cluster transitions obtained from the data. The methodology of CROM can be interpreted as a discretization approach for the Perron-Frobenius operator, which is the dual of the Koopman operator [140, 154].

Visualization of the transition matrix $\mathcal{P}$ is useful for extracting causal features for macroscopic transition phenomena – this quality is shown in Fig. 2.13 within the

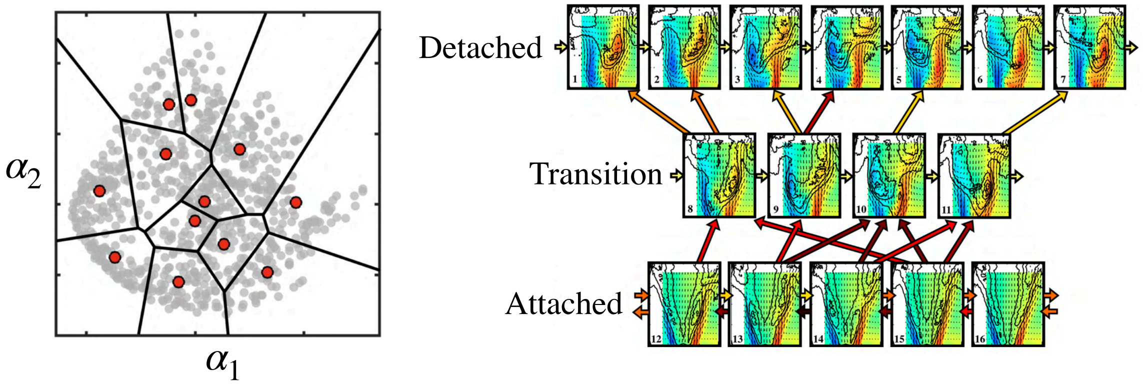


Figure 2.13: **(Left)** Visualization of Voronoi tesselation produced by K-means in a projected 2-dimensional space. Gray markers are data samples and red markers are centroids. From Ref. [19] **(Right)** Transition matrix visualization for flame transition prediction. Arrows indicate transition pathways, and brighter colored arrows mean higher probabilities. From Ref. [18].

context of flame transition prediction in gas turbine combustors [18]. In general, as an alternative to traditional modal decomposition methods like POD and DMD, CROM has been used in a wide variety other applications such as predicting vortex shedding patterns [350], cycle-to-cycle variation in internal combustion engines [47], ignition behavior in high-altitude relight applications [123], and hydrofoil cavitation [19]. The major drawbacks in this approach are (1) the selection of the number of clusters, which is equivalent to the number of modes M used in the decomposition, and (2) treating the diffusivity or low prediction-horizon times of the transition matrix $\mathcal{P}$. Different cluster numbers and observable functions (or data types) have significant effect on the predictive capability in the transition matrix. As such, significant work has been focused on providing application-dependent criteria for selecting optimal cluster numbers [18, 140, 157]. Furthermore, decompositions obtained from clustering algorithms have been used in other ROM applications alongside CROM, including (a) Galerkin projections of centroids onto the governing equations [44], (b) development of alternate transition models that improve the prediction horizon times of the original CROM method [176], and (c) utilizing other forms of clustering that address some limitations of the K-means framework, such as spectral clustering [57, 115, 320] and other variations [66, 174].

A detailed description of the K-means clustering approach – the algorithm, its advantages compared to other clustering algorithms, and applications of cluster-based decompositions on turbulent combustion problems – is delayed to Chapter III, as K-means is the backbone of the research contribution of this dissertation (see Sec. 2.6).

2.5.5 Nonlinear Projection via Autoencoders

Autoencoders are a class of neural network (NN) based models used for data compression and feature extraction. They were first introduced in *Kramer* [160], and have since then gained immense popularity in fluid dynamics applications for ROM development [41]. Within the scope of data-based ROM development for general dynamical systems, autoencoders can be thought of as nonlinear extensions (or generalizations) of POD. Instead of casting the flowfield reconstruction as a linear function of the temporal coefficients with respect to global spatial modes, the autoencoder strategy can be expressed as a nonlinear function of the coefficients as

$$\widehat{\mathbf{y}}(t) = \mathcal{D}_{\theta_{\mathcal{D}}}\left(\mathbf{a}(t)\right), \quad \mathbf{a}(t) = \mathcal{E}_{\theta_{\mathcal{E}}}(\mathbf{y}(t)). \tag{2.40}$$

In the above equation, $\mathcal{E}_{\theta_{\mathcal{E}}} : \mathbb{R}^{N_F} \mapsto \mathbb{R}^{M}$ is a nonlinear mapping function called the encoder and $\mathcal{D}_{\theta_{\mathcal{D}}} : \mathbb{R}^{M} \mapsto \mathbb{R}^{N_F}$ is the decoder. The encoder and decoder are parametrized by $\theta_{\mathcal{E}}$ and $\theta_{\mathcal{D}}$ respectively. Due to their expressive power, the functional forms of $\mathcal{E}$ and $\mathcal{D}$ are almost always neural networks of various architectures, with specific architecture type dependent on the application [113]; as such, the respective parameters represent weights and biases of the NNs. The goal of the encoder is to provide a mechanism for optimal nonlinear compression of the instantaneous flowfield into a smaller-sized vector $\mathbf{a} \in \mathbb{R}^{M}$, which is referred to as the latent vector and is analogous to the temporal coefficients in Eq. 2.29. The goal of the decoder is to recover the original flowfield from the latent vector, which is a nontrivial task since the nonlinearity and potential complexity of $\mathcal{E}$ renders inversion difficult.

The objective function in standard autoencoding applications takes the form

$$\varepsilon = \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{y}_i - \widehat{\mathbf{y}}_i\|_2^2, \tag{2.41}$$

which is a mean-squared error (MSE) between the reconstructed field and input field (N is the number of snapshots or data samples). Optimization is usually accomplished via gradient descent based on automatic differentiation [242]. As with any ROM strategy, the motivation here is order reduction – the latent vector dimensionality M must be significantly smaller than the input flowfield dimensionality for the autoencoder to be practical. A comparison with POD is provided in Fig. 2.14 – POD is recovered from the general autoencoder formulation if one casts $\mathcal{D}$ and $\mathcal{E}$ as linear operators [160].

Due to the nonlinear transformations in the encoding phase, nonlinear projection methods using autoencoders have been applied successfully in a variety of ROM frameworks that improve the predictive accuracy of POD-based models, for both nonreacting and reacting flow [169, 358]. In some applications, the task of optimizing the autoencoder parameters is performed independently of learning the latent space dynamics – in other words, after obtaining converged encoder/decoder architectures that sufficiently minimize the reconstruction objective, another ML-based modeling strategy (e.g. recurrent neural networks) can be employed to recover the dynamics of the latent space [199]. In other applications, the reconstruction errors as well as the time-evolution of the latent space can be learned in an end-to-end manner [170].

Much of the success in predictive capability for autoencoders (and data-based ROMs in general) is found in diffusion-dominated model problems [41, 100, 107], although extensions to advection-dominated scenarios [198, 199], as well as canonical

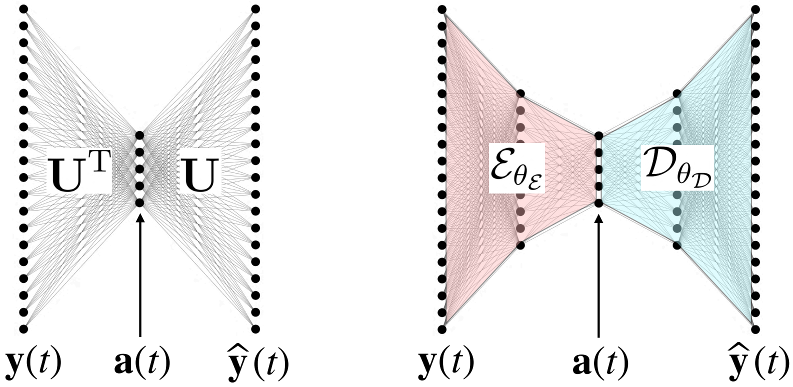

$\mathbf{U}^{\mathrm{T}}$
$\mathbf{U}$
$\mathbf{y}(t)$
$\mathbf{a}(t)$
$\widehat{\mathbf{y}}(t)$
$\mathcal{E}_{\theta_{\mathcal{E}}}$
$\mathcal{D}_{\theta_{\mathcal{D}}}$
$\mathbf{y}(t)$
$\mathbf{a}(t)$
$\widehat{\mathbf{y}}(t)$

Figure 2.14: Illustration of the basic difference between POD (left) and autoencoder-based (right) compression.

reacting flow scenarios [169], have been made in recent years. Extensions of conventional autoencoding strategies to graph-based settings (described in Ref. [113]) are promising for complex geometry treatment, although applications of these techniques for robust predictive modeling of full-geometry simulations of reacting flows (e.g. RDE or scramjet simulations) is currently out-of-reach. Overall, the architecture complexity (e.g. number of weights and biases in the encoder and decoder) and inherent nonlinearity in the transformation to the latent space is both an advantage and a disadvantage for modeling purposes. On one hand, the nonlinear transformation allows the method to capture a much greater level of complexity and feature interaction from the underlying dataset than a linear projection method (POD). On the other hand, model interpretability is lost in this approach – since the encoders readily consist of many nonlinear neural network layers, basis vector visualization methods used in POD and DMD cannot be applied. Complex fluid dynamics applications are expected to require large encoder/decoder architectures, and in turn large datasets for the training phase, to achieve respectable predictive accuracy. For this reason, research into autoencoder interpretability has gained appreciable momentum, specifically with regards to disentanglement approaches that seek to drive the encoder optimization into regions that produce uncorrelated latent space components (the goal is essentially to balance the advantages of orthogonal decomposition with nonlinear projection) [21, 38, 135].

As with POD, autoencoders should be interpreted as a general-purpose tool; their strength is revealed when they are leveraged alongside other modeling approaches. For example, nonlinear projections enabled by the encoder can enable more robust DMD modeling strategies [187], and latent spaces discovered by autoencoders can be shown to improve vanilla cluster-based methods [49]. Alternate training methods based on Bayesian inference can also be used to apply autoencoders as frameworks for generative modeling by imposing prior distributions on the latent vectors – see Refs. [152, 203] for additional details on this topic.

2.5.6 Data-based Chemistry Tabulation

In numerical combustion and reacting flow applications, a great deal of focus has been placed on accelerating detailed chemistry evaluations using data-based methods [133, 369]. Initial data-based tabulation strategies leveraging ANNs were performed by *Christo et al.* [60] and *Blasco et al.* [31] – since then, several extensions of similar ANN strategies have been applied in reacting flow solvers [20, 96, 145, 307], as well as ROM-type strategies grounded in principle component analysis [66, 192] and neural ODEs [150, 236]. Additionally, methods like in-situ adaptive tabulation (ISAT) [263] and PRISM [26] have been very successful to this end; further discussion on these approaches is delayed to Sec. 2.6.

As with physics-based tabulation methods like FPVA (Sec. 2.4.2) and CSP (Sec. 2.4.5), the motivation for data-based chemistry tabulation comes from accelerating the time-integration of the dynamical system that transports the thermochemical state ϕ for each cell in the computational domain (reproduced from Eq. 2.26 for clarity):

$$\frac{d\phi_i}{dt} = \mathbf{S}(\phi_i), \quad i = 1, \dots, N_C, \tag{2.42}$$

where ϕ is the thermochemical state vector that contains the N_S species mass fractions (or concentrations) and temperature, $\mathbf{S}(\phi)$ is the chemical source term, and N_C is the total number of cells or grid-points used in the spatial discretization. The batched ODE solve in Eq. 2.42 arises when utilizing operator splitting methods for time integration of the compressible Navier-Stokes equations (see Sec. 2.4.6). The above time integration task is problematic for two reasons: the dynamical system is highly stiff (small time steps are required in explicit integration approaches, see Fig. 1.7), and the evaluation of the instantaneous chemical source term is expensive due to the complex Arrhenius-based expression (Eq. 2.9).

Based on the above issues, the data-based chemistry tabulation effort can be divided into two application scopes: (1) accelerating the batched ODE solve for some target integration time T [31, 60, 96, 310], and (2) accelerating the instantaneous evaluation of the chemical source terms [20, 306, 307, 348]. To solidify these scopes, we define a general-purpose mapping function $\mathcal{A}$, parametrized in some way. If one wishes use data to fit the parameters of $\mathcal{A}$ to recover the state ϕ after the integration time T, the application scope becomes

$$\phi(t+T) = \mathcal{A}(\phi(t)). \tag{2.43}$$

On the other hand, if the end-goal is to fit $\mathcal{A}$ to obtain the instantaneous source terms, the application scope becomes

$$\mathbf{S}(\phi(t)) = \mathcal{A}(\phi(t)). \tag{2.44}$$

Note that the above two approaches are practically equivalent for an infinitesimally small integration time T. However, even in high-fidelity reacting flow solvers, the integration time for each cell T is often much higher than the limiting chemical timescales τ [30]. In this scenario, the modeling task in Eq. 2.43 is much more

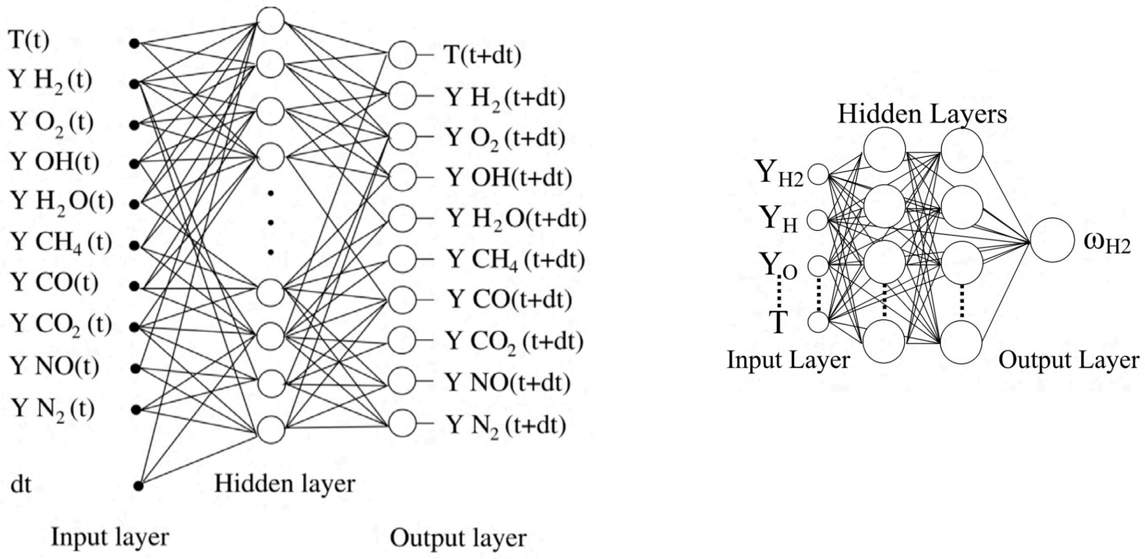


Figure 2.15: **(Left)** Deploying an ANN as a stiff time integrator (Eq. 2.43, image reproduced from Ref. [32]). **(Right)** Deploying ANN as an instantaneous source term evaluator (Eq. 2.44, image reproduced from Ref. [306]).

broad in scope than that of Eq. 2.44 – the former plays the role of a stiff time integrator (i.e. it replaces the role played by algorithms like CVODE [63]), whereas the latter plays the isolated role of providing only accelerated instantaneous source term evaluations. The differences between the two approaches is illustrated in Fig. 2.15. Although the formulation in Eq. 2.43 is much more ambitious in the sense that it replaces existing stiff time integration algorithms entirely, in some cases, limiting the modeling scope of $\mathcal{A}$ to Eq. 2.44 can be more advantageous. Often times, throwing out numerically validated stiff time integrators with provable convergence properties (such as CVODE, Newton methods, or stiff explicit solvers) is either (a) not necessary because of their optimal code implementations, or (b) too risky for production-level flow solvers. As detailed in Sec. 2.6, the research contribution of this dissertation lies in the development of a data-driven method that fits within the class of accelerating instantaneous source term evaluations (Eq. 2.44).

2.5.7 Field Transformation and Super-Resolution

Data-based models for field transformations have seen widespread use in several CFD-related applications from processes related to optical flow (e.g. using data to recover velocity fields from scalar fields [21, 75, 326]) as well as super-resolution – that is, upsampling a coarse field to a finer-resolution field for explicit closure modeling in turbulence [148]. The general-purpose mapping goal in these applications is described by

$$\mathbf{g}(\mathbf{y}) = f(\mathbf{h}(\mathbf{y})), \tag{2.45}$$

where $\mathbf{h}(\mathbf{y}) \in \mathbb{R}^{N_h}$ is an input that meaasures an observable function of the state vector $\mathbf{y}$ at some timestep (say pressure or scalar field samples), $\mathbf{g}(\mathbf{y}) \in \mathbb{R}^{N_g}$ is a simultaneously measured output observable function of the flowfield that is different from $\mathbf{g}(y)$ (say a velocity field), and f is a stationary mapping function that may be cast as a neural network or other functional form parametrized in some way. The quantity N_h represents the input dimensionality and N_g the output dimensionality. Depending on the application N_g and N_h may or may not be equal – for example, if the modeling goal for f is to recover one scalar field from another on the same grid, $N_g = N_h$. On the other hand, if the modeling goal is upsampling or super-resolution, $N_g > N_h$.

In applications related to turbulent combustion, simultaneously measured experimental diagnostic data has been used to produce cross-field transformation models via convolutional and artificial neural network based mappings functions [3, 22]. In one such application, the function $\mathbf{f}$ was trained in an optical-flow type application to produce multi-component velocity fields from reacting scalar fields at a given time instant for a model gas turbine combustor (Fig. 2.16). In another application, a convolutional neural network was trained to produce spatial chemical source term fields for sub-grid scale LES models [167]. Additionally, methods based on canonical correlation analysis [118] and singular value decompositions of cross-covariance matrices [21] have also been applied in geophysical and combustion applications to produce linear formulations of f that are less generalizable but more interpretable than nonlinear field transformations based on deep neural networks.

Super-resolution applications have been carried out extensively in recent years primarily for data-based closure modeling in large eddy simulations [99, 148]. In this class of methods, the input $\mathbf{h}(\mathbf{y})$ exists on a coarse grid that is under-resolved (i.e. the LES grid described in Sec. 2.4.1), and the output $\mathbf{g}(\mathbf{y})$ exists on a fine grid that resolves the relevant lengthscales. The goal of the super-resolution task is essentially one of enhanced interpolation or explicit filtering. In supervised approaches, the mapping function f is cast as a type of neural network (this is typically a convolutional neural network for structured grid applications, or a graph neural network for unstructured grid applications), and is optimized by fitting explicitly filtered DNS data represented as a set of stored high-fidelity simulation trajectories to the corresponding unfiltered data. This strategy has been used to reconstruct temperature fields in supersonic combustion environments to good effect [158].

Super-resolution strategies that use generative modeling frameworks have also been explored, which take a fundamentally different approach that standard field transformation methods [71]. In generative approaches, the input to the mapping function is a latent vector whose PDF is known through a prior distribution – in super-resolution applicaions for closure modeling, the latent vector may be a flowfield on an under-resolved coarse grid. Generative models parametrize the distribution of the fully resolved fields conditioned on this latent vector, e.g. via a generative adversarial networks (GAN), Gaussian processes, polynomial chaos expansions, Gaussian mixture models, among others. A training procedure then optimizes the parameters of the generative model to produce a sampling mechanism for the underlying distribution of the high-fidelity resolved fields conditioned on the unresolved fields, which

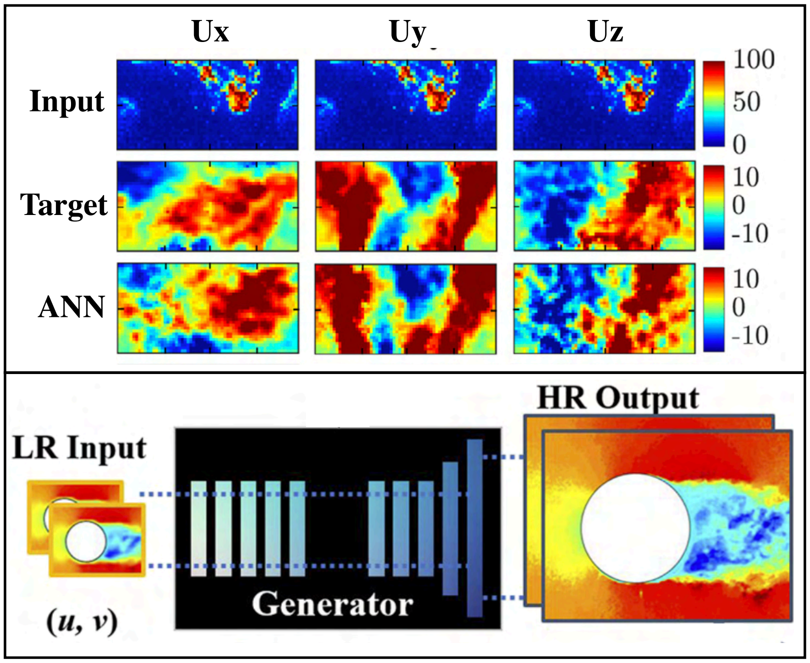


Figure 2.16: **(Top)** Applying field transformations via artifical neural networks to recover three-component planar velocity fields from an input reacting scalar field in a premixed gas turbine combustor (reproduced from Ref. [21]). **(Bottom)** Illustration of GAN-based super-resolution used for data-based turbulence closure modeling (repurposed from Ref. [71]).

effectively amounts to probabilistic field transformations. In fluid flow applications, this approach has been applied for statistical super-resolution, closure modeling, and data synthesis turbulent non-reacting flows [124] and reacting flows [34] – an example of GAN-based super-resolution used in turbelence modeling is provided in Fig. 2.16.

2.5.8 Summary and Limitations of Data-based Models

The above discussion provided background on popular data-based modeling strategies that can be used to accelerate high-fidelity reacting flow simulations. A summary of the discussed methods is provided in Tab. 2.2. The advantage with data-based approaches is that output models, if properly optimized during the training stage, are able to capture all of the complex physics contained in the training data. The ability to represent complex physics through optimal modal decomposition strategies based on space-time decoupling, neural network fits, and graph-based models among others

Modeling Approach	**Modeling / Optimization Goal**	**Class**	**Application Scope**	**Key References**
Proper orthogonal decomposition (Sec. 2.5.2)	Eq. 2.30	Unsupervised modal decomposition	Orthogonal linear projection	[29, 315]
Dynamic mode decomposition (Sec. 2.5.3)	Eq. 2.34	Unsupervised modal decomposition	Linear dynamics	[300, 301]
Cluster-based reduced order model (Sec. 2.5.4)	Eq. 2.39	Unsupervised clustering	State transition probabilities	[18, 140]
Autoencoders (Sec. 2.5.5)	Eq. 2.40	Unsupervised neural networks	Nonlinear projection	[41, 160]
Data-based chemistry tabulation (Sec. 2.5.6)	Eqs. 2.43-2.44	Supervised neural networks	Chemical time integration + source term evaluation	[60, 369]
Field transformations (Sec. 2.5.7)	Eq. 2.45	Supervised neural networks, generative models	Optical flow, sensor upsampling, coarse grid super-resolution	[22, 148]

Table 2.2: Summary of data-based models used to accelerate high-fidelity simulations of the Navier-Stokes equations

.

addresses a major limitation in physics-based modeling, namely the limiting restriction to canonical configurations.

The disadvantages in data-based approaches, however, must also be considered. A major drawback is the time-consuming training stage, which requires an expensive data collection step – populating the data matrix $\mathcal{Y}$ requires running high-fidelity simulations that are expensive to store and query. Additionally, once the data has been obtained, the optimization (training) task in itself is a primary bottleneck, as evidenced by the significant efforts undertaken to accelerate training routines like backpropagation through the development of specialized hardware. Another disadvantage is in model interpretability – at the cost of ensuring qualities like prediction accuracy and optimality with respect to the training data (i.e. black-box modeling), understanding how the models operate in data-based frameworks is a significant challenge. For example, interpreting the complex features contained in spatial modes in modal decomposition strategies, as well as isolating the way in which neural networks arrive at their decisions for tabulation approaches, can be highly nontrivial (and in some cases infeasible) tasks. Perhaps the biggest practical drawback is the tendency of data-based models to overfit the training data. If the training data is derived from a complex geometry with fixed boundary conditions, the ability of data-based models to extrapolate to different geometries (or even the same geometry with altered boundary conditions) in complex configurations like RDEs or scramjets is hindered. In this sense, the issue of non-universality present in physics-based models via constraints to canonical configurations is also present in data-based models, but at the other extreme (overfitting to complex geometries).

2.6 Research Contribution of Dissertation

In light of limitations found in both physics-based and data-based modeling strategies, the research contribution of this dissertation is focused on fusing both strategies

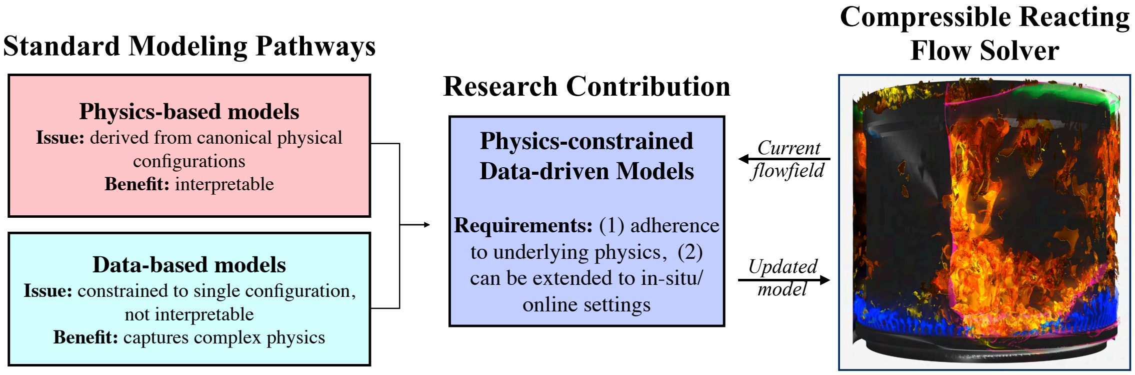

Standard Modeling Pathways
Physics-based models
Issue: derived from canonical physical configurations
Benefit: interpretable
Data-based models
Issue: constrained to single configuration, not interpretable
Benefit: captures complex physics
Research Contribution
Physics-constrained Data-driven Models
Requirements: (1) adherence to underlying physics, (2) can be extended to in-situ/ online settings
Current flowfield
Updated model
Compressible Reacting Flow Solver

Figure 2.17: Overview and scope of research contribution in light of limitations in standard modeling pathways.

together into a generalized approach termed physics-informed data-driven modeling, illustrated in Fig. 2.17. The objective here is twofold: the first goal is to augment conventional data-based modeling strategies with constraints derived from the underlying governing equations, and the second is to design these physics-informed data-driven models to be compatible with in-situ, or online, parameter adaptation. On one hand, the embedding of physical knowledge into the data-based optimization goals discussed in Sec. 2.5 eliminates the black-box nature of data-based models, and therefore allows for enhanced interpretability of predictions and confidence that the model outputs adheres to known physical trends. On the other hand, designing data-based frameworks that are able to adapt to simulation flowfields as they evolve (i.e. in-situ methods) ensures that models are not tied down to geometric configurations, and therefore improves upon the limitations in data-based and physics-based approaches related to non-universality.

The specific contribution of this dissertation is to accelerate chemical kinetics evaluations for compressible reacting flow simulations using physics-guided data-driven modeling strategy that addresses the two criteria discussed above. The approach, outlined in Fig. 2.18, utilizes a classification-based regression strategy for accelerated chemical source term estimation. In a first step, an unsupervised K-means clustering strategy is used to delineate (or classify) regions in the flow that are physically similar. Then, in a second step, localized models are deployed in each region for simulation acceleration purposes. Note that similar divide-and-conquer approaches have been explored previously for localized data-driven modeling in both reacting and non-reacting applications [80, 96, 280] – the novelty in the approach utilized here comes from (a) ensuring that the output segmentation is consistent with physical expectations in compressible reacting flow (e.g. the clusters identify meaningful regions within the detonation wave structure in RDEs), and (b) embedding physical knowledge directly into the clustering procedure itself through modifications of distance functions (termed physics-guided clustering), such that the classification identifies regions of dynamical similarity within the flowfield.

A principle modeling goal is to ensure that the flowfield delineations obtained by the algorithms adapt to the local unsteady features as the simulation evolves in time. As described in Chapters IV and V, since unsteady features of interest in compressible reacting flow like detonations are sustained by chemical reactions, the assumption is that local regions in composition space will identify coherent regions of physical similarity within the reacting flowfield. As discussed in Chapter V, appending the standard K-means optimization objective with knowledge of the governing equations results in more robust flowfield classifications that in turn produce enhanced pathways for localized combustion modeling in complex reacting flows.

2.6.1 Distinction from Related Work in Physics-Informed Modeling

The idea of embedding physical knowledge into in-situ data-based modeling frameworks for reacting flow is not new. Physics-guided modeling strategies that attempt to tackle chemistry tabulation from a reduction point of view include methods such as in-situ adaptive tabulation (ISAT) [263] and the PRISM [26] approach. In these methods, the computationally expensive numerical integration of chemical source terms (Eq. 2.43) is replaced by a look-up table. In particular, ISAT builds a trust

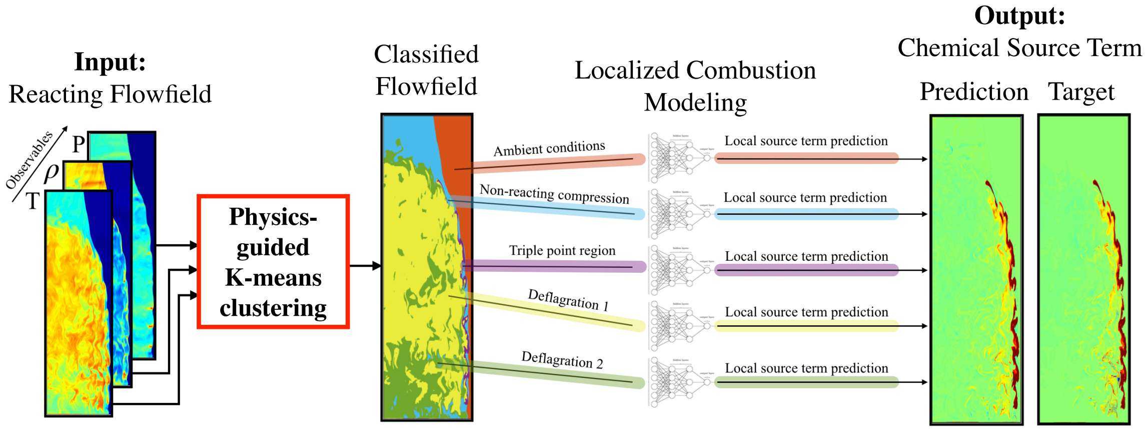


Figure 2.18: Classification-based regression strategy for chemical source term estimation.

region in thermochemical composition space using a set of ellipsoids determined by the Jacobian of the source terms, which is similar to the approach presented in Chapter V. However, in methods like ISAT and PRISM, the cost of building and accessing such tables can become expensive for large mechanisms, especially on modern high-performance computers (HPCs) that use extensive concurrency in computations to reach high throughput efficiency – the method presented in Chapter V, on the other hand, addresses these limitations by leveraging highly scalable K-means algorithms.

It should also be noted that methods sharing properties with adaptive mesh refinement (AMR) can also be interpreted as in-situ modeling frameworks that rely on data – the class of heterogeneous multiscale methods [1] and equation-free approaches [147] obtain closed-form solutions to coarse-grid residuals using online data regression strategies sourced from fine-grid simulations that may be solving a different set of governing equations valid at the target lengthscale (i.e. utilizing Boltzmann equations to feed fluxes to the Navier-Stokes equations near shocks [1]).

Additionally, the recent upheaval in physics-constrained (or physics-guided) data-driven modeling approaches in the greater CFD community has been spurred in large part by the ubiquity of supervised learning strategies based on neural networks, for which extremely fast, GPU-friendly optimization algorithms centered on backpropagation have been matured by the data science and machine learning (ML) fields. As a result, there has been a widespread adoption of various forms of so-called physics-informed neural networks (PINNs), which were brought into the mainstream in the works of *Raissi et al.* [275] and now exist as an entire field of research. The idea of the PINN framework is to regularize (or in some cases completely replace) standard data-based loss functions used in neural networks (e.g. the MSE loss described in Eq. 2.41) with residuals defined by the Navier-Stokes equations. Upon convergence in the training stage, the result is that the black-box parametrization (neural network layers) produces physically consistent outputs conditioned on the training data distribution. Although there is still a training stage here, because the physics are embedded in the loss function, much less data (or none at all [328]) is required from the training perspective to saturate the neural network parameters. Overall, the idea of embedding physics in the neural network framework is promising, though the lingering issues with the neural-network based approaches such as interpretability and quantification of error bounds are still present [289]. For example, at best, the numerics reflected by the PINNs will match those apparent in the training data, but due to the sub-optimal training procedure (the neural network parameters are initialization at random and converge to local minima), it is difficult to guarantee consistent grid convergence trends [141]. Addressing these issues remains an area of highly active research.

Alongside PINNs, physics-assisted modal decomposition strategies have also been implemented – in these methods, the modal decomposition formulation in Eq. 2.29 is modified to include time-varying basis functions, and the evolution of these basis functions can be derived from the governing equations [83, 121]. These strategies have been used as extensions to traditional data-driven ROM strategies like POD and DMD to analyze canonical turbulent reacting flows and other chaotic dynamical systems [120, 279]. It should also be noted that the idea of projecting modes derived from datasets (i.e. POD modes) onto the governing equations, as described in Fig. 2.11, can also be interpreted as a type of physics-infromed data-driven modeling approach as the governing equations are utilized in a predictive setting.

Overall, within the field of physics-informed data-driven modeling, much less attention has been paid to embedding physics in unsupervised methods based on data clustering, which as described above is the focus of this dissertation. It will be shown in Chapters III-V that physics-guided clustering strategies based on K-means can produce valuable pathways for accelerating reacting flow solvers that are markedly different from the above physics-informed data-driven modeling approaches, and improve upon the limitations discussed in Sec. 2.4.7 and Sec. 2.5.8. The objectives of the remaining chapters are as follows: Chapter III outlines the standard K-means clustering algorithm and demonstrates some applications of the method to generate ROMs for turbulent reacting flow; Chapter IV demonstrates the classification-based regression strategy for accelerating chemical kinetics evaluations described in Fig. 2.18 on high-fidelity detonation simulations; Chapter V extends the formulation in Chapter IV to include physical knowledge in the clustering objective, thereby producing delineations that are more consistent with underlying physics and combustion regimes.

CHAPTER III

K-means Clustering and Motivational Reacting Flow Applications

3.1 Introduction

Clustering methods constitute a foundational branch of data science and are used in countless applications including image compression, natural language processing, and feature extraction among others [137]. Initial clustering algorithms can be traced back to the 1930s [76]; they have since seen continued developments and breakthroughs and will undoubtedly remain as mainstays in the data science and AI literature in the future. Their application in the field of fluid dynamics – particularly for ROM development and post-processing purposes – continues to broaden in scope and robustness as the availability of high-quality, complex datasets becomes more commonplace.

Figure 3.1 displays a concise classification of the variety of algorithms in the data science literature that can be used to accomplish the general clustering goal. Although the methods are similar in that they are all clustering algorithms, each algorithm has been developed and refined by means of decades of research. As such, target application scopes of each method, and the way in which they are used to drive clustering, can vary significantly. As shown in Fig. 3.1, the approaches can be divided into two overarching categories [299]: hierarchical clustering and partitional clustering. In hierarchical clustering [215], a cluster hierarchy is formed where each level in the hierarchy represents data sample similarity at a given lengthscale (or characteristic distance) in the feature space. The primary output is a dendrogram, which is a visualization tool that allows the end-user to interpret the various levels in the hierarchy without specifying the number of clusters directly. On the other hand, partitional clustering approaches take as input the number of clusters K and assign the data to each of the K clusters using an optimization criterion [51]. The methods used in this dissertation (which are variations of K-means clustering) are categorized as partitional clustering algorithms – as such, the class of hierarchical clustering methods used in data science applications are out-of-scope here and will not be covered. Hereafter, unless specified otherwise, references to clustering operate within the context of partitional applications. It is noted, however, that there are pathways by which partitional and hierarchical approaches can be combined, i.e. utilizing partitional clustering strategies to drive hierarchical refinement procedures in a manner analogous to adaptive mesh refinement in physical space – one application of this is provided in Chapter V for flowfield classification and kinetics modeling purposes.

In general, all clustering methods are unsupervised data-based algorithms that group samples into a set of clusters, where each cluster encodes some notion of sample similarity. The primary output of the class of partitional clustering algorithms is a type of labeling mechanism that provides, for a given input sample, either (a) its corresponding cluster index represented by an integer (hard clustering), or (b) a probability mass function that encodes cluster ownership (soft clustering)[1]. The objective of the partitional clustering procedure is to ensure that samples belonging

[1] Although there is a distinction between hard and soft-clustering methods, for practical purposes, they often coincide. For example, in soft clustering methods, an additional step that extracts the ”most probable” cluster from the probability mass function is usually performed.

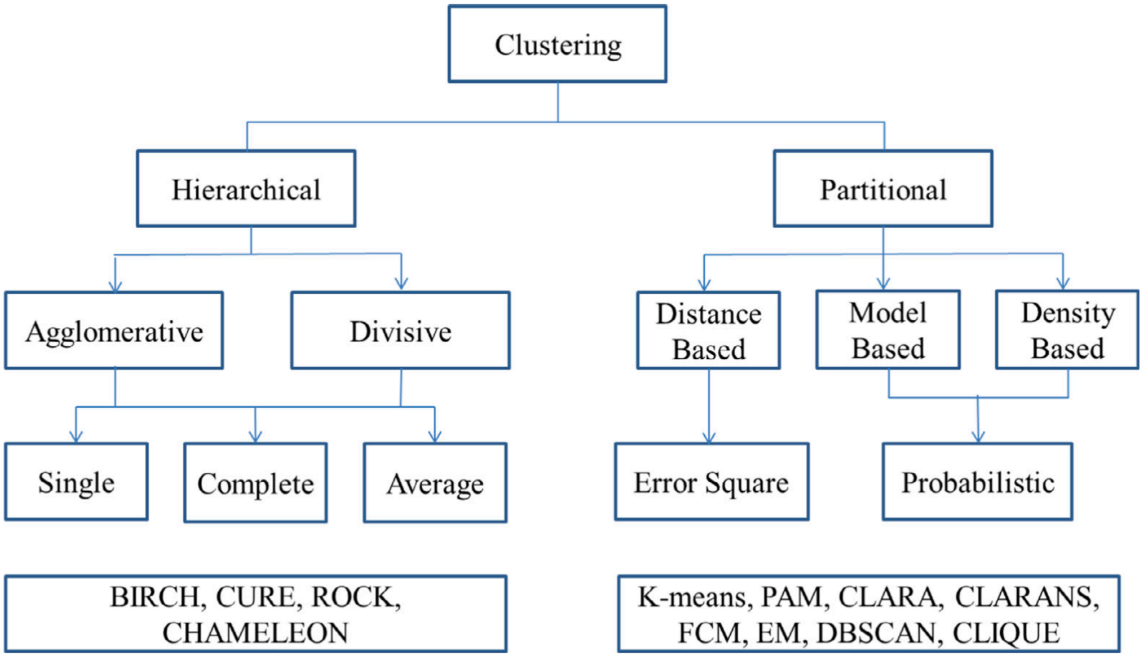


Figure 3.1: Taxonomy of data clustering algorithms reproduced from *Saxena et al.* [299]. The focus of this dissertation is on K-means clustering, which falls within the distance-based partitional category.

to the same cluster are similar and, conversely, samples belonging to different clusters are dissimilar. Quantification of the notion of similarity, which typically takes the form of a distance function in the underlying phase space (or feature space) in which each of the data samples reside, is one of the primary challenges in data clustering. Once the distance function is provided, the clustering goal can be mathematically represented as a type of optimization problem over the dataset (termed the clustering objective function) – the way in which various clustering tasks go about formulating the clustering objective based on the provided distance measure is what separates the different algorithms and approaches.

Clustering methods of all types have been applied to both non-reacting and reacting flow problems. A common application is to utilize clustering for feature extraction, thereby facilitating easier expert-guided post-processing strategies that drive physical understanding of complex flow-chemistry interactions [18, 93, 254]. Clustering strategies have been used for modeling purposes as well, where the goal is to either deploy the resulting cluster labeling mechanism within a flow solver for acceleration purposes (e.g. for cluster-guided combustion modeling [20, 33, 237, 280], which is the focus of Chapters IV and V), or to drive projection-based ROMs via data-based modal decompositions (see Sec. 2.5.1) that replace conventional CFD-based flow solvers entirely [18, 44, 140].

The goal of this chapter is to describe the basic methodology and characteristics of the K-means clustering algorithm [323], which is one of the most widely used partitional clustering algorithms in data science. As detailed in Sec. 2.6, because the K-means strategy drives the primary research contribution of this dissertation (flowfield classification for accelerating chemical kinetics evaluations), this chapter intends to inform the reader of all relevant algorithmic and model-oriented properties of K-means as a precursor to Chapters IV and V.

The remainder of the chapter is outlined as follows. Section 3.2 presents the standard K-means algorithm. Then, Sec. 3.3 outlines some of the advantages and disadvantages of the K-means clustering framework from the modeling perspective – particular emphasis is placed on comparisons with POD and spectral clustering methods due to their prevalence in combustion analysis and modeling. Sections 3.4 and 3.5 present key extensions of the K-means algorithm that allow for data-based model development in complex reacting flows. More specifically, Sec. 3.4 applies the K-means driven cluster-based reduced order modeling (CROM [140], as described in Sec. 2.5.4) strategy to predict flame transition in swirl stabilized combustors, and Sec. 3.5 uses K-means to produce an alternate modal decomposition framework referred to as time-axis clustering. These sections are included in this chapter to provide the reader some additional physical context related to the robustness of the K-means algorithm for modeling turbulent reacting flow, beyond the classification-based regression approach discussed later on in Chapters IV and V. Lastly, concluding remarks are provided in Sec. 3.6.

3.2 K-means Algorithm

The input dataset is given by the matrix $\Phi = [\phi_1, \phi_2, \ldots, \phi_N] \in \mathbb{R}^{D \times N}$, where N is the number of data samples. A sample ϕ_i in the dataset (i-th column of Φ) resides in a D-dimensional phase space[2] (i.e. $\phi_i \in \mathbb{R}^D$). Note that the source of the dataset for purposes of presenting the K-means algorithm below is left ambiguous here – the representation and significance of each of the D dimensions of the ϕ_i depends on the application, which in turn depends on the sampling procedure used to populate Φ. For example, if each of the ϕ_i represents an instantaneous flowfield snapshots derived from simulations or experimental measurements, D is the number of grid points or pixels describing the underlying numerical grid and each individual component of ϕ_i represents a measurement (i.e. velocity magnitude, pressure, etc.) at the grid point. As another example, the dataset Φ can contain samples of points in a thermochemical composition space, in which case D is equivalent to the number of species N_S and the individual components of the ϕ_i contain species concentrations or mass fractions.

The K-means procedure uses the dataset Φ to discretize the the D-dimensional phase space into finite set of K *clusters* denoted $\mathcal{K} = \{\mathbb{K}_1, \mathbb{K}_2, \ldots, \mathbb{K}_K\}$. The clusters satisfy the following four qualities: 1) $\mathbb{K}_l \subseteq \mathbb{R}^D$ for $l = 1, \ldots, K$, 2) the $\mathbb{K}_l$ are strictly disjoint and non-intersecting, 3) the union of all elements in $\mathcal{K}$ exactly reconstructs $\mathbb{R}^D$ (the clusters are space-filling), and 4) the cardinality of $\mathcal{K}$, which is the number of clusters K, must be between 1 and N. The end-result is that the clusters contained in $\mathcal{K}$ produce a centroidal Voronoi tessellation [44] of the underlying phase space in which the data samples ϕ_i reside – the K-means procedure can therefore be interpreted as a means for data-adaptive coarse-graining for $\mathbb{R}^D$, which, as seen in Sec. 3.4, is a useful property that can be leveraged for ROM development.

Given the samples $\phi_i \in \mathbb{R}^D$, the K-means procedure produces the non-intersecting

[2] ”Phase space” and ”feature space” can be used interchangeably in this context.

set of clusters in $\mathcal{K}$ by minimizing the objective function

$$E_\phi = \sum_{k=1}^{K} \sum_{i=1}^{N} \left\| \mathbf{L}_{i,k} [\phi_i - c_k] \right\|_2^2 \tag{3.1}$$

based on the initialization and convergence of a finite set of K centroids denoted $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$. The number of clusters K is one of two major inputs to the K-means algorithm, the other being the initial locations of the centroids in the D-dimensional phase space (the initialization procedure is described further below). Equation 3.1 is referred to as the within-cluster sum of squares (WCSS), and depends entirely on the centroid values. It provides a statistical scalar measure of the clustering objective based on within cluster variances about the centroids: a cluster set $\mathcal{K}$ produces a low value for the K-means objective if (a) all samples of the same cluster are similar, and (b) samples belonging to different clusters are dissimilar. As implied by Eq. 3.1, the notion of similarity (i.e. the distance function) is supplied by the Euclidean distance between sample ϕ_i and centroid c_k as

$$d(\phi_i, c_k) = \|\phi_i - c_k\|_2^2 = \sqrt{\sum_{d=1}^{D} (\phi_{i,d} - c_{k,d})^2}, \tag{3.2}$$

where $d(\cdot, \cdot)$ represents a distance evaluation and $\phi_{i,d}$ (resp. $c_{k,d}$) is the d-th component of the sample (resp. centroid).

The centroid $c_k \in \mathbb{R}^D$ is defined as the arithmetic mean of all samples within the k-th cluster as

$$c_k = \frac{\sum_{i=1}^{N} \mathbf{L}_{i,k} \phi_i}{\sum_{i=1}^{N} \mathbf{L}_{i,k}}. \tag{3.3}$$

In both Eqs. 3.1 and 3.3, the quantity $\mathbf{L}_{i,k}$ is the integer value extracted from the i-th row and k-th column of the cluster assignment matrix $\mathbf{L} \in \mathbb{Z}^{N \times K}$. In the standard K-means algorithm, for sample ϕ_i and cluster index k, the corresponding value in the cluster assignment matrix is unity if the respective sample is closest to c_k in the

Euclidean sense and zero otherwise. This can be formally expressed as

$$\mathbf{L}_{i,k} = \begin{cases} 1 & \text{if } k = \underset{j}{\operatorname{argmin}} \left\|\phi_i - c_k\right\|_2^2, \text{ and} \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

In other words, a sample ϕ_i resides in cluster $\mathbb{K}_k$ if it is closest to the corresponding centroid c_k in the Euclidean sense. Consequently, samples belonging to the same cluster are closest to the same centroid, and a sample can belong to only one cluster.

Note that the centroid is a statistical quantity that exists in the same phase space as the samples in the dataset Φ, as per Eq. 3.4. As such, the k-th centroid is interpreted as a representative (also known as a generator) of the underlying cluster or Voronoi cell $\mathbb{K}_k$. For example, if the samples ϕ_i are flowfields that can be visualized in physical space (i.e. time snapshots), the c_k centroids represent locally averaged flowfields that can also be visualized in physical space. Similarly, if the ϕ_i are thermochemical composition vectors, the centroids are "average" composition vectors or species concentrations. This quality not only allows the K-means partitions to be inherently interpretable, but also allows for the development of modeling frameworks based on treatment of the ideally small set of K centroids.

The baseline K-means algorithm is provided in Alg. 1. Given the dataset Φ, the desired number of clusters K, and initial locations for the centroids $\mathcal{C}_{old}$, the goal of the K-means algorithm is to find new centroid positions $\mathcal{C}_{new}$ that reduce the objective function in Eq. 3.1. In short, the algorithm (known as Lloyd's algorithm) consists of an iterative procedure that alternates between two steps: 1) update centroid locations via Eq. 3.3 using the cluster assignment matrix $\mathbf{L}$ from the previous iteration, and 2) update the cluster assignment matrix $\mathbf{L}$ via Eq. 3.4 using the newly updated centroids. These two steps are repeated until centroid positions converge, where convergence is defined as the point at which the centroids change by a negligibly small amount.

It can be shown that given an initial set of centroids, the above iterative procedure guarantees monotonic decrease of the objective function in Eq. 3.1, thereby producing a phase space partition $\mathcal{K}$ that minimizes within-cluster variance with respect to the input data [37].

Algorithm 1 K-means Clustering Algorithm (Lloyd's Algorithm)

```
Data:
(1) Set number of clusters $K$
(2) Set convergence error tolerance $\varepsilon_{tol}$
(3) Initialize dataset $\Phi = [\phi_1, \phi_2, \dots, \phi_N]$ to be clustered.
(4) Set initial centroids $\mathbf{C}_{old}$.
(5) Initialize convergence criterion: $\varepsilon_c \leftarrow$ Inf
(6) Set value for maximum number of iterations.
Result:
(1) Converged set of centroids $\mathbf{C}_{new}$.
(2) Cluster assignment matrix $\mathbf{L}$ (Eq. 3.4).
while $\varepsilon_c \geq \varepsilon_{tol}$ do
    a) Copy centroids: $\mathbf{C}_{new} \leftarrow \mathbf{C}_{old}$
    b) Update cluster assignment matrix using $\mathbf{C}_{old}$: $\mathbf{L} \leftarrow Eq.$ 3.4
    d) Update centroids using cluster assignment matrix: $\mathbf{C}_{new} \leftarrow Eq.$ 3.3
    e) Compute convergence criterion: $\varepsilon_c \leftarrow \|\mathbf{C}_{new} - \mathbf{C}_{old}\|_F$
    f) Break if maximum number of iterations is reached
end
```

A primary advantage of K-means the simplicity of Alg. 1, leading to fast convergence times. The complexity of K-means scales as $\mathcal{O}(NKDT)$, where T is the total number of iterations required to converge the centroids [323]. The linear scaling in number of clusters K, number of samples N and sample dimensionality D allows for a great deal of flexibility when considering parallel programming strategies to accelerate the baseline algorithm. In most data science applications, one typically observes $N \gg D$ – as such, extensions of Alg. 1 utilize parallel programming frameworks on CPUs (e.g. MPI [366] or OpenMP [216]) and GPUs (e.g. CUDA [85]) to distribute the distance evaluations and centroid updates over many cores and/or threads. An example of the linear scaling in computational cost provided by an an MPI implementation of Alg. 1 is shown in Fig. 3.2 for reference.

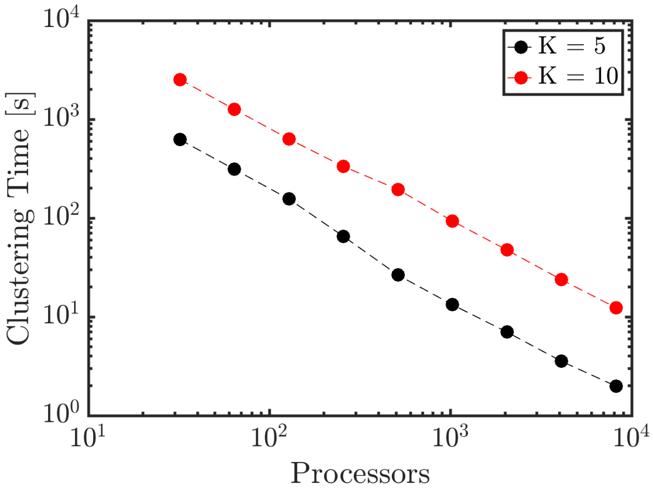


Figure 3.2: MPI implementation of Alg. 1 following the method of Ref. [72]. Dataset properties are $N = 10^8$ and $D = 20$. Results are shown for $K = 5$ (black) and $K = 10$ (red).

Centroid convergence rates are highly dependent on initial centroid locations as well as the input number of centroids, K. The naive approach is to simply initialize centroids randomly – this is ill-advised, because random initialization not only scales poorly with increase in sample dimensionality D, but also faces the risk of initializing two or more centroids close to one another in the phase space, which increases the chances of producing redundant clusters and sub-optimal values for the WCSS. Instead, an alternative approach that in practice avoids these problems is the K-means++ initialization procedure [13]. Instead of picking the initial locations arbitrarily, K-means++ initializes centroids with the goal of achieving a large amount of separation between the centroids. The procedure is summarized as follows. The first centroid, c_1, is randomly assigned to a snapshot ϕ_i in Φ. Then, c_2 is assigned to another snapshot with probability proportional to the squared distance from its closest centroid. This initialization progresses for all centroids up to c_K, and ensures that the chance of choosing an initial centroid location far from an already existing centroid is high. Note that the K-means++ initialization is stochastic, whereas the K-means algorithm itself is deterministic for a fixed initial set of centroids. Therefore, convergence of Alg. 1 using multiple realizations of starting centroid locations is necessary to ensure statistical confidence in the K-means output [18]. In general, if the dataset Φ is amenable to the clustering goal (i.e. the data is arranged in clusters suitable for the K-means procedure), the locations of the converged centroids will largely be independent of the starting positions.

Perhaps the biggest disadvantage in K-means (and almost all partitional clustering methods) is the requirement of providing K as an input to the algorithm [157]. One approach is to utilize the objective function directly to inform an optimal number of clusters. Here, the K-means algorithm is run for a wide range of K values, and the K value that results in a significant decrease in the objective function is chosen (this is referred to as the "elbow" effect and is analogous to truncation methods used to determine the number of retained modes in POD). Other approaches like the silhouette score [285] and X-means [246] formulate different measures for clustering quality as alternatives to the WCSS in Eq. 3.1 to drive cluster number selection.

It should be noted that the above scalar metrics for guiding K selection are designed for low-dimensional datasets that exhibit "true" clustering patterns, where the quality of the clustering output can be determined based on label accuracy computed using class assignments known a-priori. However, in most physics-based applications, there is no notion of cluster accuracy because there is no reason to believe a "true" clustering exists. Instead, the focus is either on (a) interpretability of the clustering output through the centroids (to guide physical analysis of flowfields, for example), or (b) the prediction accuracy of a given model constructed from the coarse-grained phase space. Since these facets are rarely considered in data science applications, conventional cluster number selection methodologies derived from the data science literature often do not work for fluid flow applications [19]. For example, in the context of flowfield classification discussed in Chapters IV and V, the clustering algorithm is not used to discover "true" clusters in a sense. It is instead used to partition the phase space in which the observed system resides in a coarse-graining procedure, thereby enabling localized modeling strategies for kinetics acceleration. In such cases, cluster number selection should be driven by optimizing application-specific modeling accuracy. Pathways to do so are provided in both Sec. 3.4 as well as in Chapter V.

3.3 Properties of K-means Clustering

K-means is useful because it allows the user to associate a single value, the centroid $\mathbf{c}_k$, to the general behavior of the cluster. In other words, in K-means it is straightforward to assign a *single* representative to the group. This is both a good and a bad thing. The good is that interpretability becomes easy, since the phase space of the centroid and what it represents (a local average) is known and accessible. The bad is that, from a statistical viewpoint, the ability to assign the mean value to a group (i.e. assigning the centroid to the cluster) is an implicit characteristic of a Gaussian density in the cluster. In fact, the K-means clusters represent multivariate Gaussian distributions, of which the centroid is the mean and the covariance is a scaled identity matrix [214, 323]. Since the covariance is a scaled identity matrix, the resulting density for each cluster is spherical, which is why K-means can only be used to produce globular clusters in whatever phase space the centroids reside.

This Gaussian interpretation comes by considering the more general case of the mixture model representation of the data probability density function (PDF)

$$f = \sum_{k=1}^{K} \alpha_k f_k, \tag{3.5}$$

where α_k is the probability of encountering cluster k, f_k denotes the the cluster-local PDF, and f is the final density function output. From Bayes rule, the posterior probability of a snapshot belonging to cluster k is

$$p(k \mid \phi_i) = \frac{p(k)p(\phi_i \mid k)}{p(\phi_i)} = \frac{\alpha_k f_k(\phi_i)}{\sum_k \alpha_k f_k(\phi_i)}. \tag{3.6}$$

If one chooses to represent the densities f_k as multivariate normal distributions with identity covariance matrices scaled by a single paramter σ, then the maximum likelihood estimates for the parameters of the normal distributions (the means and σ, which is a scalar that encodes the cluster size) using the expectation-maximization algorithm based on the above posterior lead to the K-means algorithm [214].

With this background, it is easy to identify situations in which K-means "fails". In particular, K-means clusters become unreliable when (a) the data contained within each cluster are not accurately modeled by Gaussian distributions, and (b) the within-cluster sample size is very small, such that there is little statistical confidence in the definition of the centroid. Additionally, in data-science applications, failure of a clustering algorithm qualitatively assessed by its inability to accurately identify the true clustering of some manufactured dataset using prior knowledge of the number of clusters. This can be illustrated with the famous half-moon example shown in Fig. 3.3, for which the true number of clusters is $K = 2$. K-means has clear difficulties in representing the moons with $K = 2$ – it cannot capture the concavity of the half-moons due to the globular nature of its clusters.

Many other clustering algorithms are able to capture the two half-moon clusters and the associated concavity. One example is shown in the middle plot of Fig. 3.3 for spectral clustering [345], which is a manifold-type method based on running standard K-means in a subspace spanned by eigenvectors of a graph Laplacian matrix [345]. From the results in Fig. 3.3, one might be inclined to say that a method like spectral clustering (or any other clustering approach that can pick up non-spherical shapes) is better than K-means. What is less obvious is that even though spectral clustering

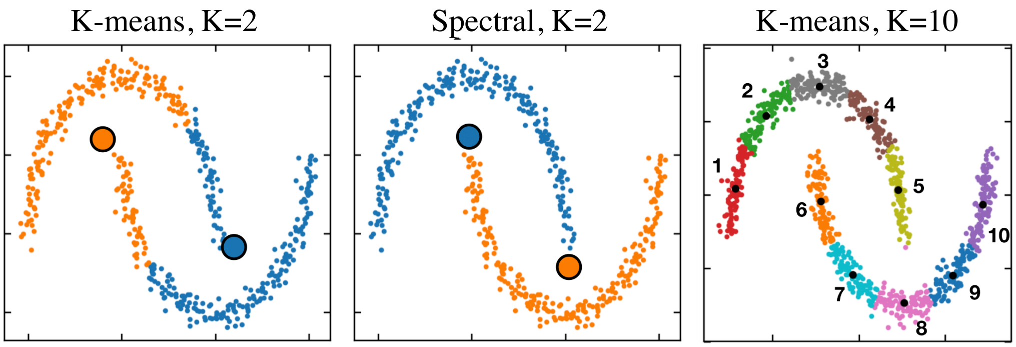


Figure 3.3: **(Left)** K-means clustering for moon dataset using $K = 2$. Small markers denote data points, large markers denote centroids. **(Middle)** Spectral clustering output for moon dataset using $K = 2$. Small markers denote data points, large markers denote locations of data belonging to same cluster. **(Right)** K-means clustering for moon dataset using $K = 10$. Colors denote cluster assignments, black markers denote centroids.

gives the true result for $K = 2$ for this example, the clusters are not space-filling in the D-dimensional phase space. In other words, complex, non-globular clusters that may be identified by other clustering algorithms (such as spectral clustering) by their nature cannot be easily characterized by a single representative point (i.e. the mean) in the same phase space as the objects being clustered. In this sense, the simplicity of the K-means clusters can be seen as both and advantage and a disadvantage.

Ultimately, the simplicity of clusters produced by K-means is desired for interpretability, but the ability to cluster complex geometries like the half-moon is required if one is to use these methods for complex fluid dynamics applications. This is because structures are much more likely to be concave in extremely high-dimensional phase spaces characterized by complex dynamical systems, where the curse of dimensionality prevents the user from representing volumes properly unless unrealistically large amounts of data are available.

A workaround is to use K-means with higher values of K. This is somewhat analogous to using a series of piecewise linear functions to approximate a single nonlinear function. As a result, the usage of K-means becomes more nuanced: for the moon example, since this amounts to requiring $K > 2$ to cluster what are truly only 2 groups, an additional *labeling* step for the K-means clusters is needed to facilitate interpretation. For example, the rightmost plot of Fig. 3.2 shows how K-means can recover the complex moon structures. However, to make any sense of the output, an additional step that assigns meaning to each cluster is required. In other words, the user must specify that in Fig. 3.2 (right), the first 5 clusters are associated with the top moon and the remaining with the bottom moon. Note that in practice, this labeling step is only required in physical analysis of the clustering output – practical utilization of the labeling step is shown in Sec. 3.4. If the goal is to use the partition as a model, interpretation of the clusters themselves become less important, and K selection is instead guided by balancing modeling accuracy with computational cost (see Chapter V).

With respect to modeling, a major benefit of K-means is the fact that the space-filling Voronoi tessellation can be thought of as a mesh consisting for the D-dimensional phase space. This is shown in Fig. 3.4. The figure displays an analogy to finite volume based adaptive mesh refinement methods (see Sec. 2.4.3) – the centroids can be represented as ”cell centers” inherently adapted to the subset of phase space occupied by the dataset Φ. As described in Chapters IV and V, equipped with this partition, tabulating expensive nonlinear functions like chemical source terms for a set of input query points in $\mathbb{R}^D$ using the centroids as proxies for average cluster behavior becomes possible.

3.3.1 Comparison with Proper Orthogonal Decomposition

A useful perspective to the minimization provided by K-means is as follows. If the centroids (here D-dimensional) are organized in the matrix $\mathbf{C} \in \mathbb{R}^{D\times K}$, then Eq. 3.1 can be re-written in matrix form as

$$E_\phi = \mathrm{tr}[(\Phi - \widehat{\Phi})(\Phi - \widehat{\Phi})^T], \tag{3.7}$$

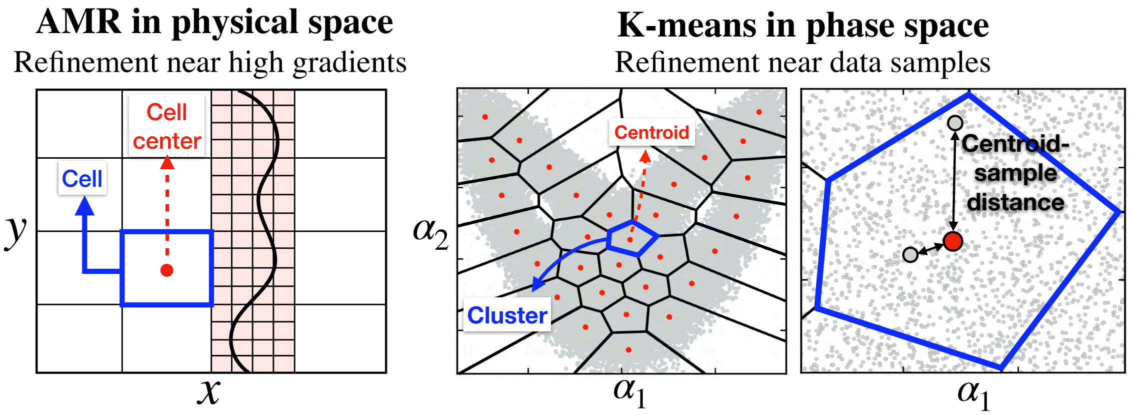


Figure 3.4: **(Left)** Illustration of adaptive mesh refinement in physical space using finite volume based discretization. Refinement occurs in regions of high flow gradients. **(Right)** 2D visualization of K-means output performed on the Lorenz system [180]. Gray points denote data samples, red points denote centroids, and black lines denote Voronoi cells. Zoom-in on one Voronoi cell is shown to emphasize variation in centroid-sample distances.

where $\widehat{\Phi}$ is the reconstructed data matrix defined as

$$\widehat{\Phi} = \mathbf{C}\widetilde{\mathbf{L}} \approx \Phi. \tag{3.8}$$

In Eq. 3.8, $\widetilde{\mathbf{L}}$ is the matrix $\mathbf{L}$ from Eq. 3.4 whose columns are divided by the corresponding number of samples in the cluster. Given the above formulation of the K-means objective, an interesting parallel to the objective of proper orthogonal decomposition (see Sec. 2.5.2) can be made, which seeks to minimize the Frobenius norm of the error due to data reconstruction as follows (reproduced from Eq. 2.30 for clarity):

$$E_{POD} = \|\Phi - \widehat{\Phi}_{POD}\|_F = \sqrt{\mathrm{tr}[(\Phi - \widehat{\Phi}_{POD})(\Phi - \widehat{\Phi}_{POD})^T]}. \tag{3.9}$$

In Eq. 3.9, $\widehat{\Phi}_{POD}$ is the reconstructed data matrix facilitated by the reduced set of POD modes. As implied by the similarity between the corresponding objectives (Eq. 3.7 and Eq. 3.9), K-means can be interpreted as a type of discrete version of POD in that it optimizes the reconstruction of Φ using K basis vectors (the centroids) with the additional constraint that the corresponding K clusters are space-filling and

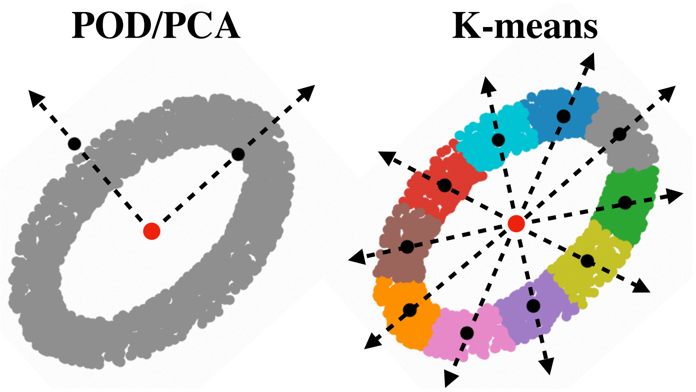


Figure 3.5: **(Left)** POD modes (dashed arrows and black markers) for a simple 2d ellipse dataset (gray markers). Center of ellipse is given by red marker. **(Right)** K-means clustering ($K = 10$) on the same dataset. Colors indicate cluster labels and centroids are given by dashed arrows and black markers.

non-overlapping in D-dimensional phase space. This is formally shown in [74].

Both K-means and POD rely intrinsically on the covariance of the error matrix $(\Phi - \widehat{\Phi})(\Phi - \widehat{\Phi})^T$; however, the optimization approaches for each method produce modes that are fundamentally different within the modal decomposition framework (see Sec. 2.5.1). To illustrate the basic differences, a comparison between K-means centroids and POD modes is shown for a simple ellipse-derived two-dimensional dataset in Fig. 3.5. It can be shown that optimal data reconstruction, which is the goal of POD, requires the POD modes to be recovered from the eigenvectors of the data covariance matrix, thereby producing an orthonormal basis. On the other hand, the goal of K-means centroids is to minimize within-cluster variation in the data. The disadvantage in the centroids is that they can be redundant depending on the value of K (or sub-optimal from the reconstruction perspective), but the advantage is that they provide an accessible coarse-grained representation of the underlying phase space, which allows for the development of interpretable modal decomposition and tabulation strategies for modeling reacting flows.

3.4 Cluster-Based Analysis and Prediction of Flame Transition in Gas Turbine Combustors

3.4.1 Background

The issue of flame stabilization is critical for lean premixed flames with the perspective of controlling pollutant formation and ensuring robustness at various operating loads [10]. In swirl-stabilized combustors, the interaction of multiple recirculation zones with flame propagation impacts the stabilization process. Two different stabilization mechanisms are feasible [61, 329, 332]: a) a shear layer stabilized flame resulting in a flame attachment to some feature of the geometry, and b) lifted flames that are stabilized near the stagnation surface formed between inflowing gases and an inner recirculation zone. For practical considerations, the transition into a lifted state might serve as a precursor for flame blow-off [324]. Prior studies have focused on elucidating physical mechanisms that cause transition between the attached and lifted flame states, which are due to either inflow or other operational variations. The focus of this work is to use one such experimental study to develop a prognostic data-driven model with a twofold purpose: 1) to understand how the flame instability occurs and 2) to predict the flame transition.

Instabilities in complex systems such as gas turbines are subject to multi-scale mechanisms that require a simplified representation to be meaningful. A first technique is to quantify the growth rates of perturbations, which are obtained using system identification (SI) methods [223]. The perturbations are quantified for some quantity of interest, e.g. pressure at some location. The SI technique can be input-output based, where the response of the combustor to perturbations are used, or output based [223, 224, 262]. A second technique utilizes data decomposition to obtain a simplified representation of the governing equations. Proper orthogonal decomposition (POD) [29, 322] projects the governing equations onto some meaningful flow modes, and dynamic mode decomposition (DMD) attempts to linearize the governing equations [300]. Such methods determine instability modes from experimental data and the variation of their relative contribution to the flow field. These characteristic modes can be used to directly analyze physical system properties of interest, and also to construct a reduced-order model (ROM) of the system. As described in Chapter II, these approaches have been used in the past for combustion chambers [11, 211], and additional studies have utilized laser diagnostics to determine characteristic modes [112, 229].

From a prognostic viewpoint, the above methods are derived from two different processes. SI and DMD-related techniques are purely data-driven approaches, where finite measurements of the system are used to construct empirical tools which may then be used to predict combustor state transitions. While such tools are easier to use in a practical setting, interpreting the model in terms of the physical processes is not straightforward. Further, DMD related methods invoke linearization of the underlying dynamics, which may not be valid in many practical configurations. In contrast, POD-type techniques attempt to simplify the theoretical governing equations to predict the future states of the system. First, data from the system is used to construct lower-order representations, or a ROM; second, this model can be propagated in time in order to predict the future state of the system. Since the decomposition techniques can be related to physical processes, the first step provides meaningful projection operators which can be used to analyze the physics of the combustion. However, the projection applied to the governing equations leads to issues with closure and numerical discretization, similar to those encountered in reduced-fidelity models such as RANS/LES [265, 266, 276].

Recently, a so-called cluster-based reduced-order modeling (CROM) framework built on K-means clustering has been proposed by *Kaiser et al.* [140], and was described briefly in Sec. 2.5.4. Unlike prior decomposition tools that obtain a linear operator to describe the nonlinear dynamics, CROM retains the nonlinearity of the underlying system. Furthermore, CROM avoids a direct projection of the governing equations, thereby alleviating the closure problem; instead, it constructs a discrete-time Markov process from a set of experimental or high-resolution simulation data. As a result, CROM provides a physically-meaningful decomposition of the dataset which allows to understand from the available data how the flame instability occurs [122]. It also provides a probabilistic model for the forecast of the combustor state. In the original work of *Kaiser et al.* [140], CROM was used to construct the model of a turbulent mixing layer. This method has also been used to predict cycle-to-cycle variations in internal combustion engines via cluster discretization of data [47] and cavitation mode transitions for model hydrofoils [19].

In this study, CROM is used to develop a means for transition mechanism analysis and to model flame topology transition in a swirl combustor. Experimental stereoscopic particle image velocimetry (S-PIV) and OH planar laser induced fluorescence (PLIF) images are used to develop a predictive ROM for flame transition. The flow modes obtained are analyzed to illustrate the physical information that can be obtained from the CROM methodology. The remainder of this paper is organized as follows: experimental data and operating conditions are provided in Sec. 3.4.2; the CROM methodology is discussed in Sec. 3.4.3; outcomes of the CROM approach are used to gain physical intuition about the flame transition mechanism in Sec. 3.4.5; results of the CROM approach both in terms of the prediction horizon time and forecast capability for different data types are discussed in Sec. 3.4.6. Finally, conclusions and future directions are presented in Sec. 3.4.7.

3.4.2 Experimental Configuration and Dataset

The data used for this analysis was acquired in the experiments of Ref. [8], which contains additional details on the combustor and diagnostic techniques. A brief summary is provided here.

Figure 3.6a shows the gas turbine model combustor, which has been used for a number of previous studies, both experimental [50, 202, 231] and numerical [58]. A review addressing progress and key findings in experimental studies on hydrodynamic instabilities, including additional perspective on the advancements in numerical simulations of instabilities in swirling combustors is provided in Ref. [132]. The schematic shown in Figure 3.6c depicts the type of flame shape geometry observed in the attached and lifted states as seen by this particular swirl combustor.

Premixed fuel and air are fed through a plenum to the radial swirler before entering the combustion chamber, where vortex breakdown generates a strong inner recirculation zone. In the present case, a fuel-air mixture of equivalence ratio of $\phi = 0.60$ is fed to the combustor at a preheated temperature of 400 K. The fuel is made of 80% CH_4 and 20% CO_2 by volume. The air flow rate is 400 SLPM. This case was selected from the test matrix in Ref. [8] because it exhibits a high number of transitions between clearly defined attached and detached flame states (8 observed in a span of 1.5s), while operating at fixed equivalence ratio and flow rates. Thus, the flow conditions were not changed to force a transition, and the inherent system can be considered ergodic; the flame experienced intermittent and spontaneous (in the sense of being apparently random in time) transitions between the detached and lifted states.

Data was collected using 10 kHz repetition-rate OH PLIF and S-PIV, providing simultaneous time-resolved 2D measurements of the OH radical distribution and of the three velocity components over a time span of 1.5 s. Figure 3.6b shows typical instantaneous OH PLIF images in attached and detached states. In the attached state, high OH concentrations are present in the shear layers that separate the inner and outer recirculation zones, while in the detached state, a rotating helical vortex core generates a highly asymmetric OH field. Attached and detached flame configuration shapes are shown in the drawings in Fig. 3.6c – the flame in the attached state takes on the characteristic V-shape, whereas the M-shape is observed in the detached. There were roughly 8 total transitions captured in the 1.5 s dataset (combining attached-to-detached transitions and the detached-to-attached transitions). For the specified operating conditions, the flow-through time is 14.6 ms (from Ref. [8]) and the flame transition timescale (determined by manually observing the 8 detached-to-attached and attached-to-detached transition times in the OH-PLIF images) is on the order of 10 ms.

In order to reduce the computational overhead, the PLIF images were reduced in size from 832 × 504 pixels to 104 × 63 pixels via nearest-neighbor interpolation based filtering, which preserves the large-scale flame features. The effective resolution of the down-sampled OH PLIF was 0.71 mm. The full-resolution PIV data was used (79 × 53), with a vector spacing of 0.78 mm and interrogation box size of 1.56 mm. Note that in Fig. 3.6a, the PIV window is not perfectly symmetric about the combustor centerline, which is reflected accordingly in the figures and discussion below. Nevertheless, the procedure used here only requires the data to coincide in time. Different datasets spanning different parts of the combustors can therefore be used.

3.4.3 Cluster-Based ROM Methodology

In this section, the CROM methodology is described. Essentially, it processes time-resolved data of any kind, identifies recurrent patterns in the system dynamics, and creates a probabilistic predictive model for these patterns. Here, the data are the 2D experimental OH-PLIF and PIV images. The set of experimental images is denoted by the data matrix $\Phi = [\phi_1, \ldots, \phi_N] \in \mathbb{R}^{D \times N}$, where N is the number of snapshots available and each subscript denotes a different snapshot. The snapshots are ordered in time and are contained in the columns of Φ. This time-ordered quality

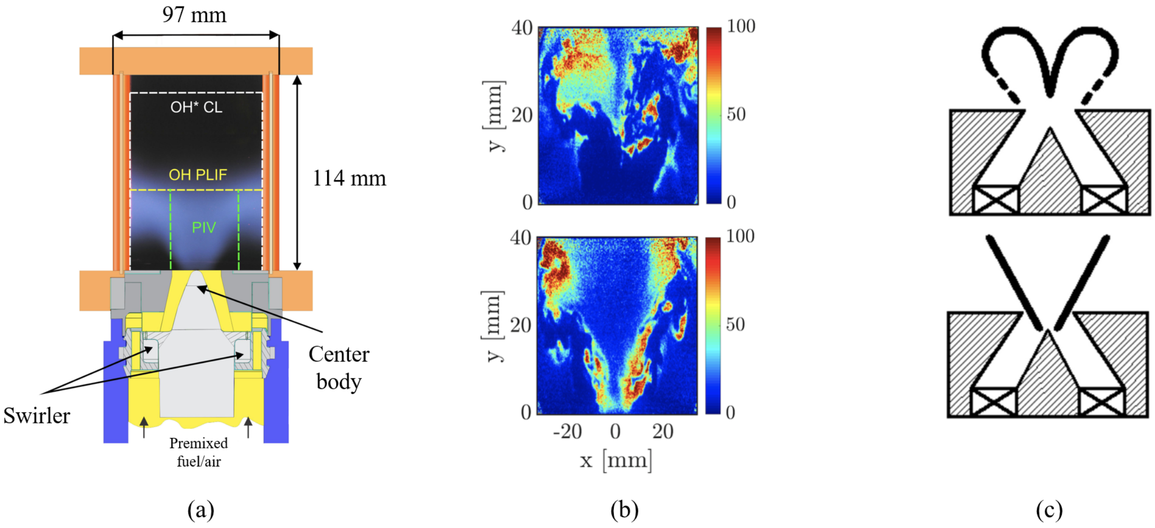


Figure 3.6: (a) DLR combustor schematic. The fuel consists of 80% CH_4 and 20% CO_2 by volume. (b) OH-PLIF snapshots of detached (top) and attached (bottom) flames in units of relative pixel intensity. (c) Sketches of M-shaped detached flame (top) and V-shaped attached flame (bottom) (from Ref. [10]).

is important to create the predictive model, but not important when classifying the data. Each column ϕ_i in Φ is a vector of pixel values associated with PIV and/or OH-PLIF images, and is of size D (i.e, $\phi_i \in \mathbb{R}^D$). Snapshots are separated by a time-step Δt that is determined by the 10 kHz sampling rate identical across all measurement types. In this section, the data classification (clustering) and the probabilistic model (transition matrix) obtained from the CROM methodology are described. Further, a procedure to relate the cluster outputs to the individual swirler states is also presented (a systematic labeling of the clusters as detached flame, attached flame or transitioning flame).

3.4.3.1 Utilization of K-means Clustering

The first step of CROM is to map the set of snapshots Φ to a smaller set of *centroids* $\mathcal{C} = \{c_1, c_2, \cdots, c_K\}$ using the K-means procedure described in Sec. 3.2, where $c_i \in \mathbb{R}^D$. In this application, because the input dataset Φ contains snapshots made of D pixels, the centroids are also images made of D pixels that represent some

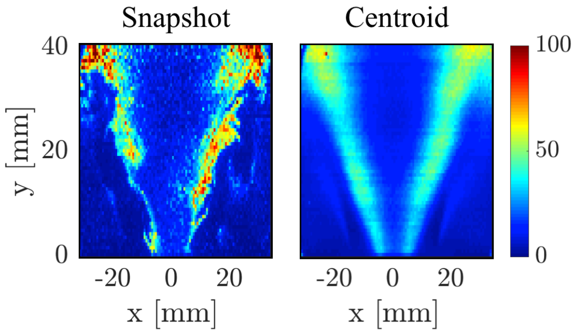


Figure 3.7: An example of a PLIF snapshot of an attached flame with its corresponding centroid.

pattern of the flow field. Centroids in CROM can be interpreted as delta functions that discretize the probability density function (PDF) of the states. States nearby one another are represented by the same delta function. This step is reminiscent of classification methods used in language interpretation: the same word can be pronounced differently, but a language interpreter finds common characteristics of these different signals to assign them the same meaning.

Depending on the flow pattern represented by each centroid, the average distance of the snapshots assigned to each centroid can vary. Ideally, each centroid c_i should be as close as possible to its assigned snapshots such that the centroid actually represents these snapshots. Note that the centroids are not flow fields that can be observed, but statistical patterns that approximate what can be observed. An illustration of this statistical effect is shown in Fig. 3.7, which juxtaposes an example PLIF snapshot with its corresponding centroid.

3.4.3.2 Transition Matrix

The prediction tool of the CROM approach is the transition matrix, which is the practical ROM necessary to make state forecasts. The transition matrix provides the probability of a snapshot transitioning from one cluster to another within a given forward time-step Δt. In a mathematical sense, this matrix represents a Markovian discrete time-step mapping. The transition probability of an image in cluster j

transitioning to an image in cluster k is obtained based on the association matrix $\mathbf{L}$ as:

$$\mathcal{P}_{j,k} = \frac{\sum_{m=1}^{N-1} \mathbf{L}_{m,j}\mathbf{L}_{m+1,k}}{\sum_{m=1}^{N-1} \mathbf{L}_{m,j}}, \text{ for } k = 1, \ ..., \ K. \tag{4}$$

The transition matrix $\mathcal{P}$ defined above is valid only for a finite time-step Δt. A key assumption regarding the cluster-based ROM framework is that of Markovianity, i.e., that the finite time-step transport of probability distributions using the transition matrix generated by the snapshots in Φ is memoryless. Any finite-dimensional spatial discretization of Markovian governing equations will naturally yield a Markovian dynamical system. Here, it has only been assumed that the governing equations for the field measured are themselves memoryless. This is reasonable since fluid flow equations are memoryless. Moreover, given the large density of data points, it is reasonable to consider the system experimentally observed to follow Markovian dynamics as well. Model validation techniques for forecasting purposes are presented in Sec. 3.4.6. Additional details are provided in Ref. [140].

Additionally, before centroid analysis and implementation of the transition matrix $\mathcal{P}$ in a prediction setting, it is useful to re-arrange the clusters in some probability-based order. One method (used in these results) is to order the clusters by descending eigenvalue modulus of $\mathcal{P}$ for clearer identification of cluster groups within the matrix structure [140].

3.4.3.3 Forward Propagation Model

The centroid set $\mathcal{C}$ combined with the transition matrix $\mathcal{P}$ now serve as the prognostic model. From an initial PDF of the clusters $P_0 \in \mathbb{R}^K$ the forward model determines the probability distribution of centroids at a future time t:

$$P_t = \mathcal{P}^{n_s} P_0, \tag{5}$$

where $n_s = t/\Delta t$. It is important to note that in the infinite time limit, the probability distribution represents the statistically stationary state that can be obtained from all the snapshots. This is the ergodic behavior of the transition matrix:

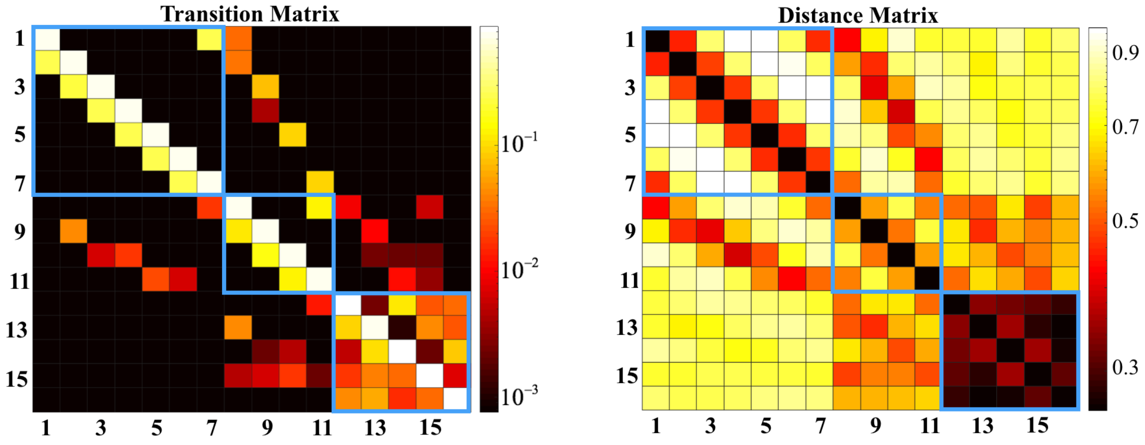
$$\lim_{t\to\infty} P_t = \boldsymbol{e} \approx \boldsymbol{q}, \;\; q_k = \frac{\sum_{m=1}^{N} T_{m,k}}{N}, \text{ for } k = 1, \;..., \; K. \tag{6}$$

In Eq. 6, $\boldsymbol{e}$ represents the PDF in the ergodic limit of the transition matrix, and $\boldsymbol{q}$ represents the initial cluster distribution of the snapshots. Furthermore, as $\mathcal{P}$ is a Markov matrix, its first eigenvector $\boldsymbol{v}_1$ corresponds to the distribution $\boldsymbol{e}$ ($\boldsymbol{e} = \boldsymbol{v}_1$). Simply put, this feature states that the model gradually loses its predictive capability as $\mathcal{P}$ is raised to successively higher powers. Therefore, the transition matrix does not give information about any upcoming flame transition or flame state after a certain finite number of time-steps. This point can be defined as a finite critical time, τ_h, after which the probability distribution remains stationary regardless of the initial PDF P_0; this is referred to as the *prediction horizon time* [140]. As will be discussed in the results section, τ_h is highly sensitive to the type of data used to apply the CROM methodology, and is a key metric in evaluating the forecast power of one particular dataset over another. This will be the object of the discussion in Sec. 3.4.6.

3.4.4 Labeling of Centroids

The clusters obtained from K-means isolate flow patterns in an unsupervised manner, but need to be assigned to states of interest for meaningful interpretation and forecasts. Here, it is explained how each cluster is labeled with the “detached flame”, “attached flame” or “transitioning flame” category. The centroid classification procedure is presented using a model generated with a combined dataset (OH-PLIF + PIV-x/y/z components).

Figure 3.8a displays the transition matrix with probabilities appearing as elements



(a)

(b)

Figure 3.8: (a) Transition matrix with substructures boxed. (b) Distance matrix with substructures boxed.

of the heatmap color-coded in log scale for $K = 16$. Three distinct structures are identified within the transition matrix as indicated by the blue boxes; further detail regarding the assignment of each structure to a physical state follows this list:

1. Clusters 1-7 represent a periodic step-like probability structure, where the most likely path (aside from remaining in the same cluster) is to move on to the next cluster.

2. Clusters 12-16 represent a highly interconnected probability structure (i.e. a snapshot in one cluster in this group has a roughly equally probable chance of moving on to any other cluster in this group, not just the next cluster).

3. Clusters 8-11 are also step-like and periodic, similar to 1-7, but are probabilistically connected to both of the regions described in (1) and (2).

Centroid groups are further identified via the *cluster distance matrix* shown in Fig. 3.8b, which displays the L2 distance between each centroid [140]. The distance matrix is symmetric with diagonal equal to zero.

Centroids corresponding to the attached state are expected to be relatively similar in the phase space, as there should be far fewer possible phase space realizations of an attached flame in the given domain than a detached flame. This coincides with centroids 12-16 as indicated by the distance matrix, which is a grouping of centroids markedly close together. A clear cluster grouping is also visible when observing centroids 12-16 in the transition matrix – the probabilistic interconnection of these centroids is related directly to their phase space similarity. Based on the above analysis, centroids 12-16 are associated with the attached flame state. Visualizing these centroids confirms the classification, as seen in Fig.3.11.

Of the two remaining groups (1-7 and 8-11), one must be assigned a "detached" label and the other a "transition" label. Centroids 8-11 in the transition matrix are connected to the other two groups: for example, a snapshot starting in clusters 8-11 has some probability of entering either clusters 1-7 or clusters 12-16. Centroids 8-11 are also identifiable as transition centroids via the distance matrix; centroids 1-7 are far from centroids 12-16, but centroids 8-11 are closer to both. Clusters 8-11 are therefore assigned the transition label. The corresponding centroids are plotted below in Fig. 3.13.

This leaves the detached label assignment for clusters 1-7. These clusters differ from the transition centroids since the matrix $\mathcal{P}$ does not allow transition to the attached centroids (12-16). Centroids 1-7 are plotted in Fig. 3.13, confirming the detached assignment. Note that there are more detached clusters than there are attached clusters, as is expected – the detached flame state should occupy a larger region of the phase space than the attached state based on the experimental dataset.

CROM then provides the labeled centroids and the transition matrix, which can be used for a) analysis of the centroids and transition mechanism (Sec. 3.4.5), and b) prediction of flame transition and analysis of horizon time (Sec. 3.4.6). In context of the current experimental setup, this approach is valid only when the system is

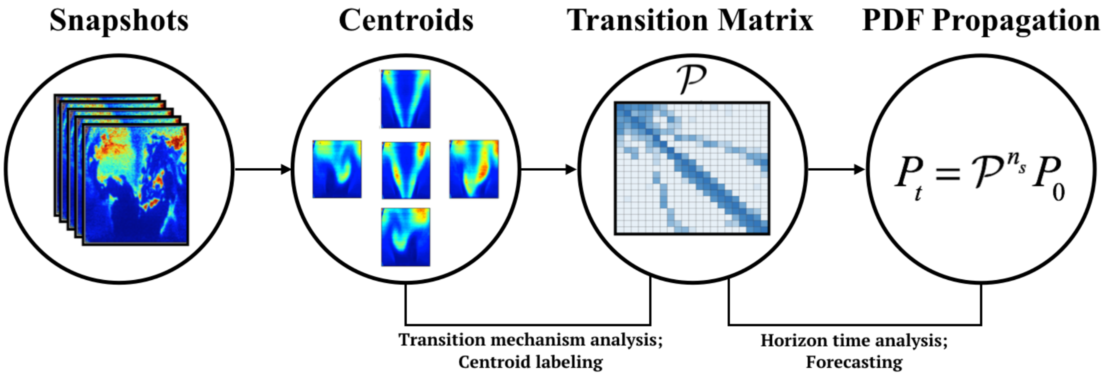


Figure 3.9: CROM workflow. The K-means clustering procedure occurs between the first two steps. Centroids and transition matrix shown here are arbitrary.

ergodic. For additional perspective, a summary schematic of the model procedure is shown in Fig. 3.9.

3.4.5 Analysis of the Transition Mechanism

Not only does the CROM methodology allow for the construction of a predictive model, but it also helps infer the physical mechanism through which particular events happen. In this case, CROM can be used to extract the sequence of events leading to flame detachment or attachment. The outcomes of CROM (centroids and transition matrix) are used in this section to perform this analysis.

First, an ideal cluster number K is chosen such that it is low enough to simplify the analysis but high enough to better identify all three flame states (Sec. 3.4.5.1). Second, the clustering process is applied to the full dataset combining OH-PLIF and all the velocity components. As a result, each centroid obtained is composed of 4 different fields (OH and the three velocity directions). It will therefore be possible to describe the interaction between the flame and the velocity field during the flame transition process. Third, this transition process (both from detached to attached flame and attached to detached flame) is analyzed by combining the information provided by the transition matrix in the form of cluster transition probabilities with the centroids for the combined dataset. In particular, the most likely sequence of events leading to attachment or detachment will be extracted.

3.4.5.1 Number of Clusters

For optimal mechanism analysis, it is critical to choose a cluster number that discretizes the phase space in such a way that all states of interest are captured. In this case, the goal is to choose a lower bound on K such that the attached, detached, and transition flame states are properly refined. Transition matrix structures and cluster time series are compared in Fig. 3.10. The *cluster time series* plots (bottom row in Fig. 3.10) show the time evolution of the images represented by associating each snapshot to its closest centroid. As seen from the data, the flame is initially in one state, but quickly transitions to another. Later in time (around $t = 0.12$s), the combustor undergoes another transition and moves back to the original state. The red highlighted regions in Fig 3.10 show this second transition – as the total number of clusters is increased, so too is the refinement of the transition mechanism. The number of clusters chosen is $K = 16$, as it best captures the transition process and contains the necessary resolution of all three flame states of interest. The transition matrix structure shows clear groups that can be labeled as explained in Sec. 3.4.4 – clusters 1-3 are associated with an attached flame, clusters 8-16 with detached, and clusters 4-7 with a transition state. Figure 3.10 also implies that the selection of K is more complex than assigning simply one cluster to one state (i.e. for three states, $K = 3$), as the cluster number is dependent on the time spent in each state. Because the K requirement depends on the sufficient resolution of the transition process, and the flame transition between attached/detached states occurs very quickly compared to the flame residence time in either attached or lifted states, in the end of the clustering process there should be more than one cluster per label. The transition matrix then has non-zero transition probability for several cluster changes, which allows variations in the density of trajectories in each individual flame state to be

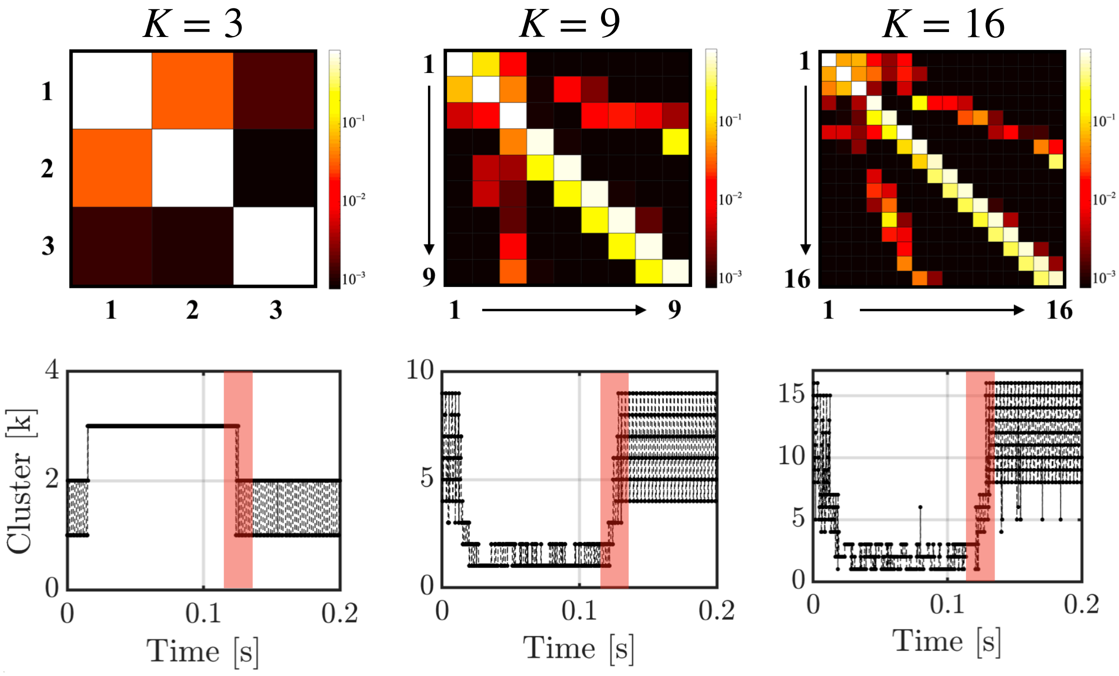


Figure 3.10: (Top) Transition matrices for $K = 3$, 9, and 16. (Bottom) Corresponding cluster time series for the first 0.2 seconds.

better represented. For example, having only one centroid in the detached state would be a drastic oversimplification of the evolution of trajectories corresponding to a detached flame, and would not capture the associated periodicity of the flame anchoring point.

Besides facilitating the interpretation of the CROM output, using the smallest possible cluster number that captures the states of interest decreases the statistical uncertainty for the entries of the transition matrix. Therefore, it should be noted that if the sole interest is the forecasting power of the model, the optimal number of clusters K will be different as will be shown in Sec. 3.4.6. For the purposes of centroid analysis as it pertains to transition mechanism identification, in the following discussion, centroids were produced for a cluster discretization of $K = 16$.

3.4.5.2 Description of the Bistable State via Centroids

In this section, the centroids obtained for the value $K = 16$ are analyzed to illuminate the behavior of the swirling flame during the attached, detached and transitioning phases.

The complete set of attached, transition, and detached centroids are shown in Figs. 3.11, 3.12, and 3.13 respectively for the full dataset. These images essentially show how the clustering algorithm organizes the raw snapshots. In the figures, OH isocontours are shown as black lines, and the arrows represent the velocity in the axial and transverse directions (x and y). Out-of-plane velocity (z component) is shown by the colored contour.

In the attached state, full flame symmetry is observed for centroids 15 and 16 which is expected – the OH isocontours indicate a similar flame shape to the instantaneous PLIF attached image seen in Fig. 3.6. However, in some attached centroids (12, 13, 14), there exists asymmetric alternating high-OH concentration regions (circled in red Fig. 3.11). For the same centroids, some slight velocity field asymmetries are also observed (”bending” of the out-of-plane velocity streaks and alternating recirculation zone presence), potentially indicating the initial formation of a precessing vortex core (PVC). This is also observed in experimental results [10, 230].

Images corresponding to the transition centroids (8-11, Fig. 3.12) show more complex structures for the velocity and OH profiles. These centroids show strong flame and flow asymmetry, suggesting that the PVC gained in strength. All transition centroids clearly display alternating recirculation zones (zig-zag structure as observed in [230]) with coinciding high OH-concentration fields. It appears that the vortices creating vorticity normal to the measurement plane serve as flame anchoring points. The transition centroids obtained can be associated in pairs 8/10 and 9/11 which exhibit symmetry with respect to the swirling axis. The radial symmetry implies that the detachment process happens independently of the swirling process, but is

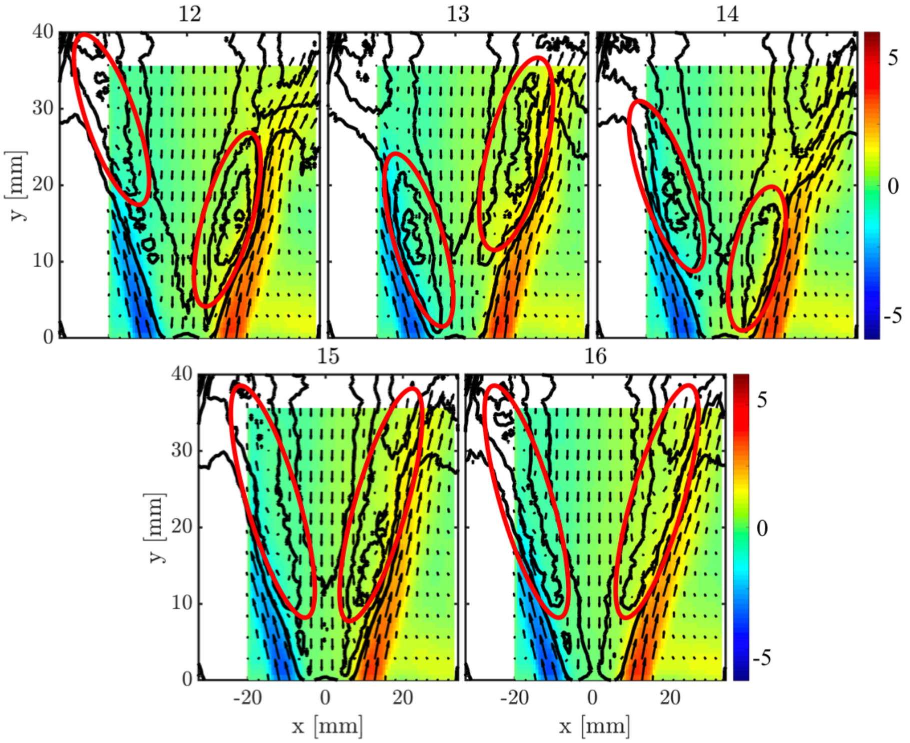


Figure 3.11: Attached centroids of the combined dataset. OH-PLIF isocontours indicated in black lines. PIV-x and y components given by arrow overlays, and PIV-z is given by the heatmap (colorbar units in m/s). Circled regions enclose regions of increased OH-concentration.

localized in the azimuthal direction.

The detached centroids are shown in Fig. 3.13. The OH concentration and the flow field are similar to those of the transition centroids, and are more pronounced. The flame front and the recirculation zones are in particular more lifted. One key difference that can be noted in terms of the flame topology is the existence of hook-like structures for the OH contour in centroid 3 and 7 (red arrows). This suggests that the flame is exposed to stronger strain rates than during the transition.

A key aspect from the above discussion is that a pure phase-space clustering process is able to isolate important flow features relevant to the transition phenomenon in an unsupervised manner.

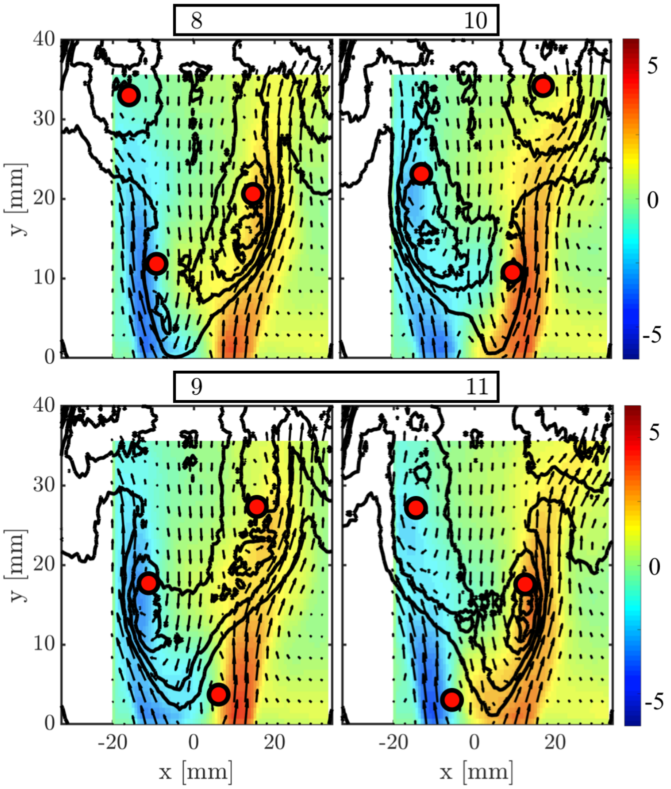


Figure 3.12: Transition centroids of the combined dataset. OH-PLIF isocontours indicated in black lines. PIV-x and y components given by arrow overlays, and PIV-z is given by the heatmap (colorbar units in m/s). Recirculation zone centers in indicated by markers.

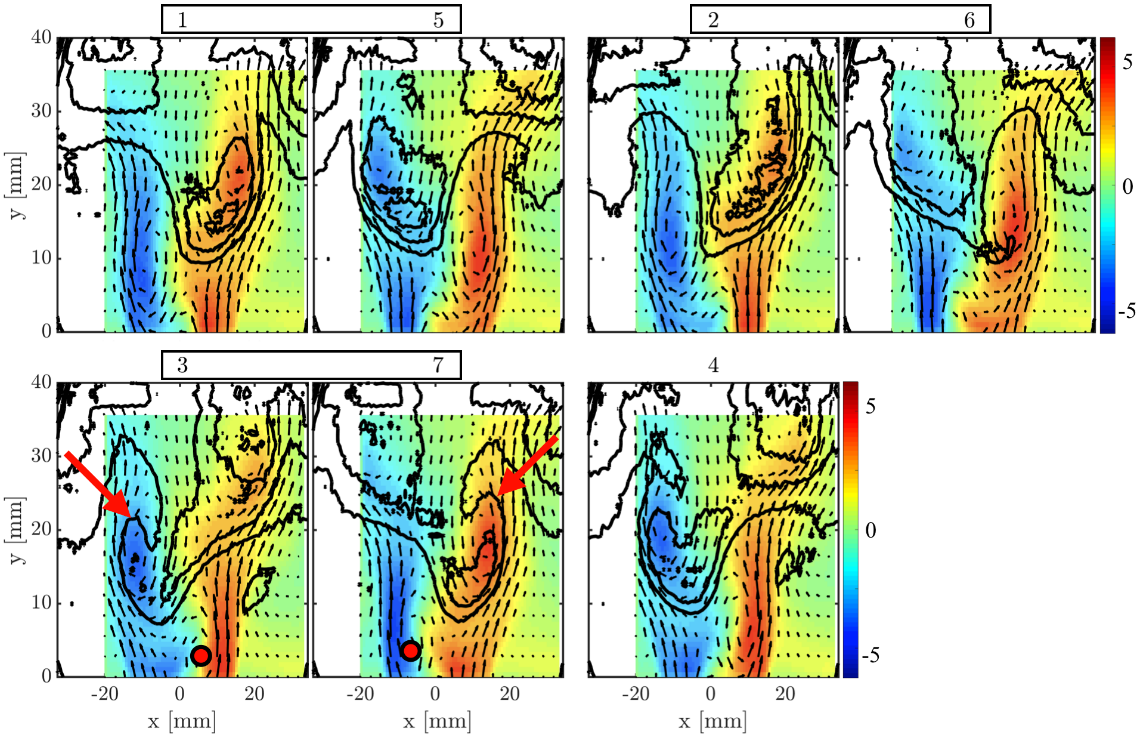


Figure 3.13: Detached centroids of the combined dataset. OH-PLIF isocontours indicated in black lines. PIV-x and y components given by arrow overlays, and PIV-z is given by the heatmap (colorbar units in m/s). In centroids 3 and 7, the hook-like structures are indicated by red arrows and the recirculation zones near the burner exit are marked in red.

3.4.5.3 Analysis of the Bistable Transition via Transition Matrix

In this section, two different transition mechanisms are analyzed: the detachment process (attached to detached flame) and the attachment process (detached to attached flame). This section combines the physical information gained from individual centroid analysis (Sec 3.4.5.2) with the probability transition information from the transition matrix in Fig. 3.8a in order to interpret the dynamics of the transition mechanism. The analysis below is conducted for model generated with the full dataset (OH and all three velocity field components).

Flame Detachment Process: Figure 3.14 displays the probability paths as outputted by the model for detachment. Colors associated with the arrows are coded in the same scale as the probabilities in Fig. 3.8a – lighter colored arrows indicate higher probabilities. In Fig. 3.14, an interesting distinction is that the attached centroids associated with slight asymmetry in the OH field (centroids 12, 13, 14 and 15) have a higher chance of moving on to the transition clusters. Interestingly, the attached centroid that is most symmetric (centroid 16) has zero chance of moving on directly to a transition centroid, implying that asymmetry is a leading indicator of detachment. The velocity field asymmetry was also used as a marker of the beginning of flame detachment in the study of Oberleithner et al. [230]. Here, the results suggest that slight asymmetry of the flame itself could be at the inception of the detachment. Further analysis will be required to clearly identify the causes of the detachment. Nevertheless, this finding highlights the capabilities of CROM in assisting the interpretation of experimental data for transient flows.

The transition centroids are characterized by the tendency to evolve into a neighboring centroid. Each cluster appears to resolve a particular phase of the periodic swirling motion. Centroid 8 tends to evolve into 9, 9 into 10, 10 into 11, and 11 back into 8, etc. In the case of a detachment, both centroids 10 and 11 are likely to evolve into the detached centroids 5 and 7. Centroids 8 and 9 can each evolve into different detached centroids: 1/2 and 3/4 respectively. Similar features are seen within transition centroid pairs in the evolution to the lifted state; for example, centroids 9 and 11 (one mirrored pair) each evolve into detached centroids which exhibit a hook-like feature on the same side of the symmetry plane. Furthermore, centroids 8 and 10 (the other transition centroid mirrored pair) evolve into lifted centroids which depict the flame on the opposing side of the symmetry plane (i.e. centroid 8 is left-leaning and evolves into a right-leaning lifted flame – vice-versa for centroid 10). The detached centroids depict a similar periodic tendency as the transition centroids, as the most probable path is to simply evolve into the next centroid. Assessment of the lower stagnation point in the detached regime gives insight to the PVC structure location, and it can be seen that this feature oscillates back and forth across the symmetry plane. An interesting feature to note is that the detachment process does not include centroid 6, which may be a product of the chosen cluster number.

Flame Attachment Process: In the flame attachment case as outlined in Fig. 3.15, it is seen that a snapshot in the first cluster has no chance to directly enter the transition region. The most likely detached-to-transition pathway is given by the progression from centroid 2 (right-leaning detached flame) to centroid 9. Centroid 2 differs from the other detached centroids in that there exists a strong recirculation zone on the left of the image (highlighted in red) almost at the same axial location as the flame anchoring point on the right. The flame can therefore be entrained toward the nozzle by the local negative axial velocity.

It is noted that centroids 1 and 2 exhibit a very similar flame front but have a very different velocity field. Centroid 1 does not play a direct role in the attachment mechanism while centroid 2 is key for the inception of the attachment. The velocity field is therefore not only at the root of the flame detachment, but also plays a key role in flame attachment. In the case that centroid 2 evolves into centroid 9, development of a recirculation zone very near the nozzle exit is seen. In fact, higher

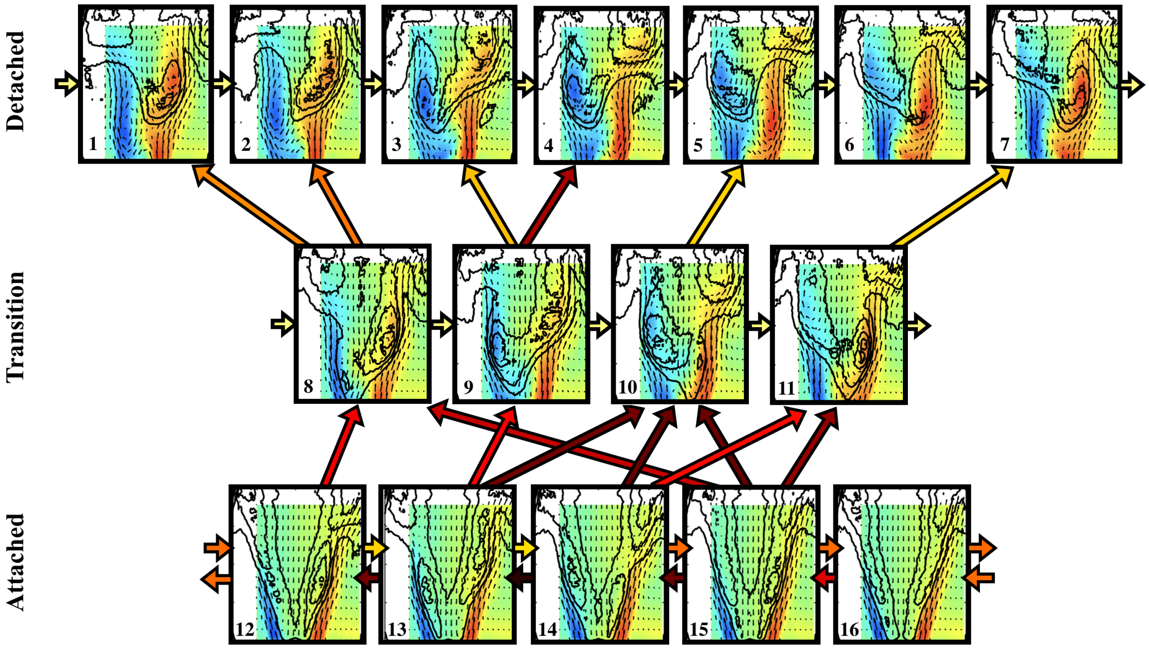


Figure 3.14: Probability paths for the cluster transitions in the flame detachment process. Arrows indicate the paths and are color-coded with the same colorbar as the transition matrix; darker colors are smaller probabilities, and brighter colors are higher probabilities.

probability cases in which a flame enters the transition state (e.g. centroids 4 to 10, 5 to 11) all depict the presence of a lower recirculation zone near the nozzle. The most probable path for transition-to-attached is the progression from centroid 8 to centroid 13. Centroid 8 is the most symmetric centroid of all the transition clusters in terms of both the velocity and OH concentration. This appears to help the flame stabilize into the "V-shape" profile characteristic of the attached configuration.

3.4.6 Prediction of Flame Transition

Along with providing information about the transition mechanism, the transition matrix can also be used as a forecasting tool. The quantification of model predictability (or predictive strength) is found in the prediction horizon time, τ_h, defined in Sec. 3.4.3. It is the time after which the transition matrix converges to the ergodic probability distribution $\boldsymbol{e}$. This convergence is not a sudden process,

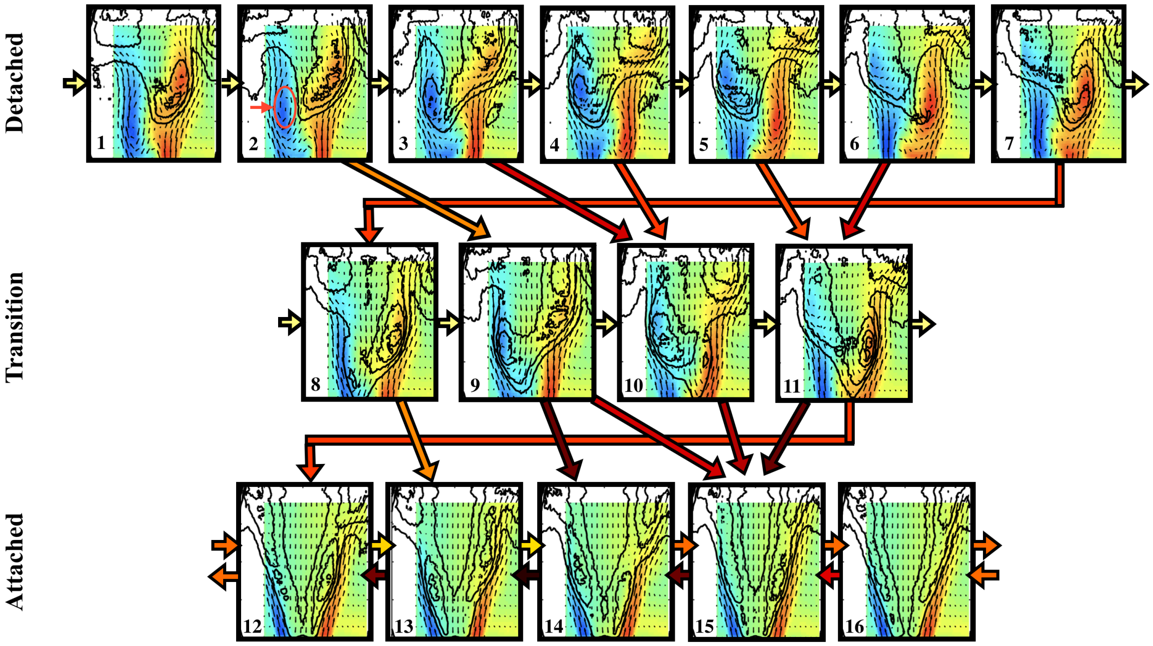


Figure 3.15: Probability paths for the cluster transitions in the flame attachment process. Arrows indicate the paths and are color-coded with the same colorbar as the transition matrix; darker colors are smaller probabilities, and brighter colors are higher probabilities.

since the transition matrix continuously diffuses towards its stationary state as it is raised to successively higher powers. Therefore, any input probability distribution will eventually converge to $\boldsymbol{e}$ given a large enough number of time-steps.

Similar to Sec. 3.4.5, it is necessary to choose a value for parameter K, the number of clusters. Here, K is chosen to optimize *forecasting or predictive purposes.* With different constraints than in Sec. 3.4.5, it will be seen that the optimal number of cluster will also be different. It should be noted that in this section, the primary focus is to compare the predictive power of different types of data, where the data types in this case are OH-PLIF, PIV-x, y, and z. This means that unlike in Sec. 3.4.5, which utilized a single model composed of a combined dataset to assess the physical mechanisms of the transition, different models will be created for each type of data to compare horizon times, uncertainty and predictive capability.

This section is organized in the following manner. First, the process of extracting the horizon time τ_h from the transition matrix is demonstrated in Sec. 3.4.6.1. Then, the procedure used to set the optimal number of clusters is described in Sec. 3.4.6.2. In Sec. 3.4.6.3, horizon times obtained with different datasets are compared. Finally, in Sec. 3.4.6.4, flame transition forecasts are compared across models constructed with different data-types (PIV measurements and OH-PLIF).

3.4.6.1 Determination of Prediction Horizon Time

The prediction capability of the transition matrix can be quantified by the prediction horizon time τ_h (see Sec. 3.4.3). There are several methods outlined in Ref. [140] that can be used to obtain τ_h for a given transition matrix $\mathcal{P}$ (second eigenvalue convergence to zero, probability distribution convergence to ergodic distribution, convergence of finite-time Lyapunov exponent, and Kullback-Leibler entropy convergence); here, the eigenvalue spectrum of $\mathcal{P}^n$ is monitored as n increases. An inherent property of $\mathcal{P}$ is that the largest eigenvalue is unity and is stationary (does not change in time), meaning that an eigendecomposition of the transition matrix raised to any discrete positive power always yields a maximum eigenvalue of one. The corresponding eigenvector of the unity eigenvalue is the statistically stationary state (the ergodic limit). As the system evolves in time, the remaining $K - 1$ eigenvalues gradually converge zero (this is a property of Markov chains). The second largest eigenvalue λ_2 is the last eigenvalue to reach zero. Its value is tracked as the transition matrix is raised to higher and higher powers. Figure 3.16 shows a typical evolution of the second-largest eigenvalue with respect to time. In practice, the prediction horizon time is defined as the time at which $d\lambda_2/dt < \varepsilon$. Here, $\varepsilon = 1\text{e}{-5}$.

3.4.6.2 Number of Clusters

Similar to what was done for the analysis of the transition mechanism in Sec. 3.4.5, the cluster number must be carefully chosen. In this section, the purpose of CROM

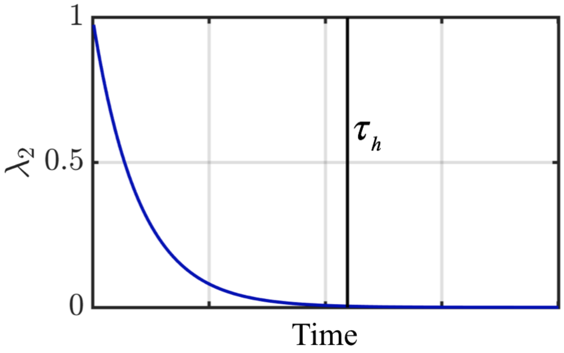


Figure 3.16: Example of second-largest eigenvalue convergence for some $\mathcal{P}$. The horizon time, τ_h, is the time at which the slope of λ_2 falls below some threshold ε.

is different than in Sec. 3.4.5, and as such the optimal number of clusters/centroids K is determined differently. Here, the goal is to optimize the prediction horizon time τ_h while decreasing the statistical uncertainty of the transition matrix entries. The measure of this statistical uncertainty is explained below.

Consider the square transition matrix $\mathcal{P}$, where each entry of the matrix is $\mathcal{P}_{i,j}$ and $i, j = 1,\ ...,\ K$. One can then associate a relative error to each element $\mathcal{P}_{i,j}$ as

$$\delta\mathcal{P}_{i,j} = \sqrt{\frac{1-\mathcal{P}_{i,j}}{N\mathcal{P}_{i,j}}}, \tag{7}$$

where N is the total number of snapshots in the dataset. The measure chosen for the uncertainty quantity is the Frobenius norm of the relative uncertainty matrix,

$$||\delta\mathcal{P}|| = \sqrt{\frac{1}{N}\sum_{i=1}^{K}\sum_{j=1}^{K}\frac{1-\mathcal{P}_{i,j}}{\mathcal{P}_{i,j}}}. \tag{8}$$

In the rest of the section, the transition matrices and the clusters are constructed with data in the range [0s, 1s] of the available measurement. This training set covers 4 of the 8 flame transitions available in the data. The last two transitions (range [1s, 1.5s]) are kept to test the model performances (testing set). Transition matrices are constructed for different number of clusters in order to find the optimal K sought

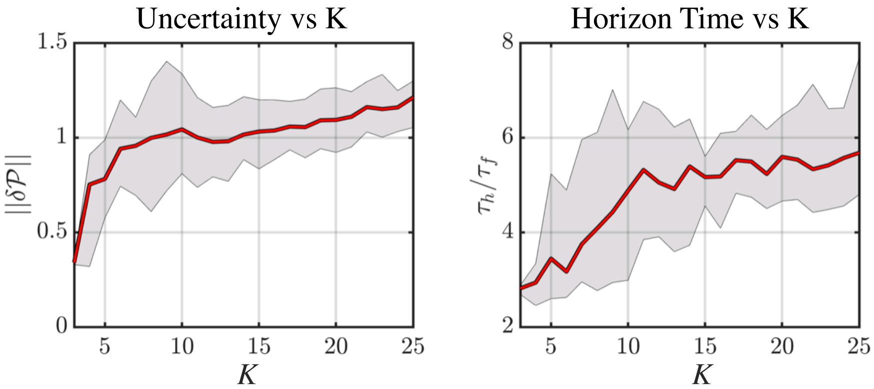
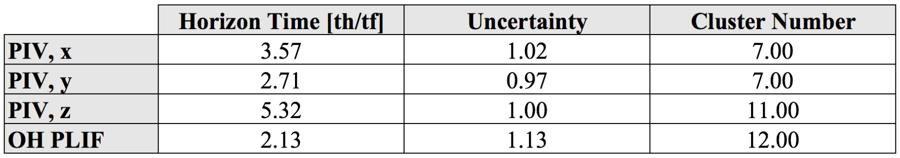


Figure 3.17: Out-of-plane velocity field transition matrix uncertainties (left) and horizon times (right) versus K. Maximum and minimum bounds are indicated by shaded boundaries derived from individual K-means++ runs, red lines indicate mean.

here.

Figure 3.17 displays the uncertainty measure and normalized prediction horizon time as a function of cluster number, K, for the PIV-z (out-of-plane velocity) dataset. Prediction horizon time τ_h was normalized with the combustor flow-through time $\tau_f = 14.6$ ms. Because the K-means++ algorithm is stochastic (see Sec. 3.4.3), 20 realizations were run to ensure that the conclusions are meaningful. The enclosed shaded regions in Fig. 3.17 represent maximum and minimum boundaries obtained from individual runs of the clustering algorithm, and the red line indicates the average of all runs.

The uncertainty metric growth rate in Fig. 3.17 decreases dramatically after an initial increase until $K = 6$. The error metric appears to be increasing, albeit slowly, beyond a small relative maximum at $K = 10$ after the sharp increase at lower cluster numbers. A similar sharp increase trend is seen in the normalized prediction horizon plot, where the mean stabilizes after $K = 10 \sim 11$. The results of both the prediction horizon and the matrix uncertainty suggest that the optimal cluster number for the PIV-z dataset could be $K = 11$.



	Horizon Time [th/tf]	Uncertainty	Cluster Number
PIV, x	3.57	1.02	7.00
PIV, y	2.71	0.97	7.00
PIV, z	5.32	1.00	11.00
OH PLIF	2.13	1.13	12.00

Table 3.1: Optimal K values with corresponding uncertainties and horizon times as extracted from Fig.3.18.

3.4.6.3 Horizon Time Comparisons

The process of choosing an optimal cluster number based on horizon time and transition matrix uncertainty was conducted for the other data-types in the same manner as shown for the PIV-z dataset in Sec. 3.4.6.2. In the end of the process, different optimal cluster numbers K are found for different data types. The results are summarized in Table 3.1. Similar model validation test trends as shown for PIV-z in the Appendix were observed for models derived from all data-types.

Figure 3.18 shows a more detailed comparisons of normalized horizon times and uncertainty measures for the four different datasets (OH-PLIF, PIV-x, y, and z directions) as a function of cluster number – this is the same plot as presented in Fig. 3.17, but with overlays for all data types for clearer trend visualization. The curves shown are average quantities of 20 runs for each data type with minimum and maximum bounds for each curve displayed in the same corresponding color.

The plot shows that the out-of-plane component of velocity (PIV-z) provides the highest horizon time for nearly all cluster numbers. In contrast, the OH-PLIF dataset interestingly gives the lowest horizon time for most of the tested cluster numbers. This is counter-intuitive since the OH concentration is a marker of the flame position and can be expected to be a good descriptor of the lift-off or attached states. This finding can be interpreted by considering the cause and effect of flame transition. The OH signal could be an effect of the flame transitioning to a new state, while the out-of-plane velocity field could be a cause. As a results, the PIV-z data contains more detail about the future state of the combustor. This agrees with prior analyses in

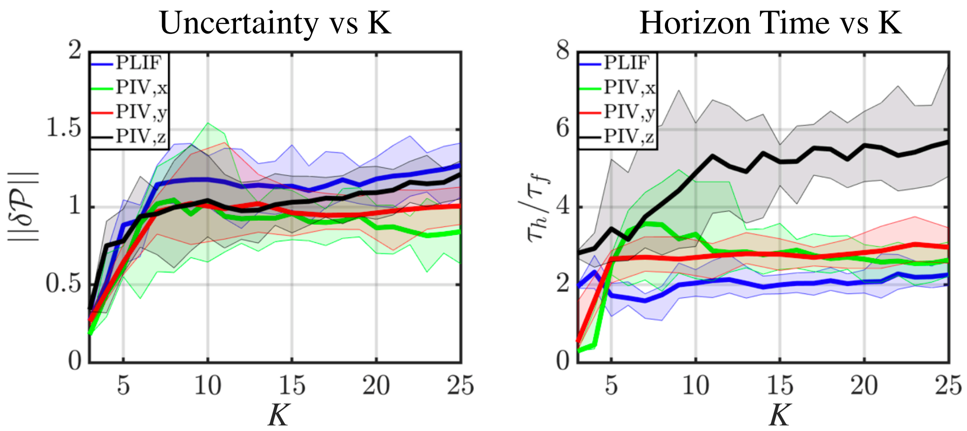


Figure 3.18: Comparisons of transition matrix uncertainty (left) and horizon times (right) for the four tested data types. The curves are averaged quantities from 20 independent K-means++ runs on the dataset. Maximum and minimum bounds are indicated by shaded boundaries.

that the PIV-z data was actually found to be at the root of flame transitions [9, 10]. The flame lift-off, for example, is aided by high-strain rates in the inner recirculation zone, which causes flame extinction and formation of a PVC. Since a signature of the PVC can be found in the out-of-plane velocity component (PIV-z), it is unsurprising that the predictive power of this field is high compared to other measurements.

3.4.6.4 Forward State Predictions

In practice, a tool to forecast transitions between different macroscopic states is invaluable to anticipate and control flame instabilities. As discussed in Sec. 3.4.3, CROM can be used to predict the future probability distribution of the combustor state in terms of the clusters. Using the cluster labeling method described in Sec. 3.4.4, the predictions can be formulated in terms of the macroscopic flame states. Starting from a state where the flame is in either the detached or attached state, the goal is to understand if the model can accurately predict the probability of exiting the initial state, or transitioning. In other words, the objective is to determine how accurately the model can predict a flame detachment (flame exiting the attached state) as well as a flame attachment (flame exiting the detached state). The data was partitioned in training and testing sets in the same way as indicated in Sec. 3.4.6.2. The state prediction process is conducted as follows:

1. Pick one cluster labeled as attached (resp. detached). Start with a $\delta-$distribution P_0 in this particular cluster. P_0 is a vector equal to 0 everywhere and 1 for the index of the cluster picked. This situation would correspond to observing the flame as being in a state that corresponds to the cluster picked.

2. Determine all snapshots associated with the chosen attached (resp. detached) centroid in (a) in the cluster time series for the testing dataset.

3. Advance P_0 to P_{τ_p} using the transition matrix (Eq. 5), where τ_p is the time in the future at which predictions are sought.

4. Use the experimental snapshots at time τ_p relative to the original snapshots determined in (b) to determine an experimentally obtained $P_{\tau_p,exp}$ from the testing dataset.

The computed (from model, $P_{\tau_p,exp}$) and experimental (from data, P_{τ_p}) probabilities are then compared to determine the accuracy of the ROM. This is schematically shown in Fig. 3.19.

Forecasts are compared for out-of-plane velocity field (PIV-z) and OH-PLIF measurements for conciseness. The number of cluster used for each data-type is indicated in Table 3.1. Two prediction scenarios were tested for each data-type: 1) the snapshot initially resides in the attached state (100% chance of a snapshot in attached cluster), and 2) the snapshot initially resides in the detached state (100% chance of a snapshot in detached cluster). For each scenario, three prediction times were assigned to illustrate model performance at various time scales: $\tau_p = \tau_f = 14.6$ ms (one flow-through time), $\tau_p = 0.1\tau_f = 1.46$ ms (short-time prediction), and $\tau_p = 7.0\tau_f = 10.22$ ms (long-time prediction, i.e. beyond the prediction horizons for both PIV-z and OH-PLIF).

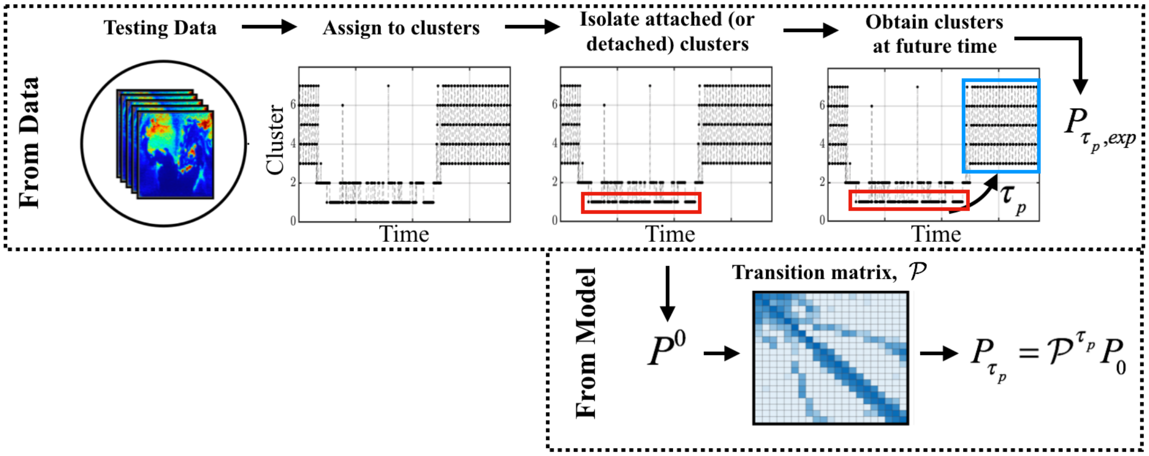

From Data
Testing Data
Assign to clusters
Isolate attached (or detached) clusters
Obtain clusters at future time
Cluster
Time
Time
Time
τ_p
$P_{\tau_p,exp}$
From Model
P^0
Transition matrix, $\mathcal{P}$
$P_{\tau_p} = \mathcal{P}^{\tau_p} P_0$

Figure 3.19: Schematic juxtaposing the procedure for finding $P_{\tau_p,exp}$ (directly from data) and the procedure for for finding P_{τ_p} (from CROM).

The initial cluster distributions P_0 were propagated through the transition matrix for these specified τ_p values, and the resulting final distribution was then compared with that of the testing dataset.

Because different models are generated when using PIV-z and PLIF data (different cluster numbers, different clusters, different transition matrix), the comparison of performance between datasets is non-trivial. Here, the initial cluster for each dataset is chosen such that 1) the cluster appears to depict a similar flow state, and 2) the final probability distributions $P_{\tau_p,exp}$ are similar for both model outputs. This ensures that both clusters are representative of a similar combustor state. Therefore, the test compares the ability of both datasets to predict transitions that are as close as possible. This criterion becomes more apparent in the results below. This method was used with different pairs of clusters showing similar results. Here it is shown for one of these pairs.

Detachment prediction results for $\tau_p = \tau_f$ are shown in Fig. 3.20. Note that final probabilities are in terms of two categories: 1) ”attached” and 2) ”not-attached”. The category ”attached” is essentially a persistence probability for remaining in the attached state after time τ_p. The category ”not-attached” includes probability of

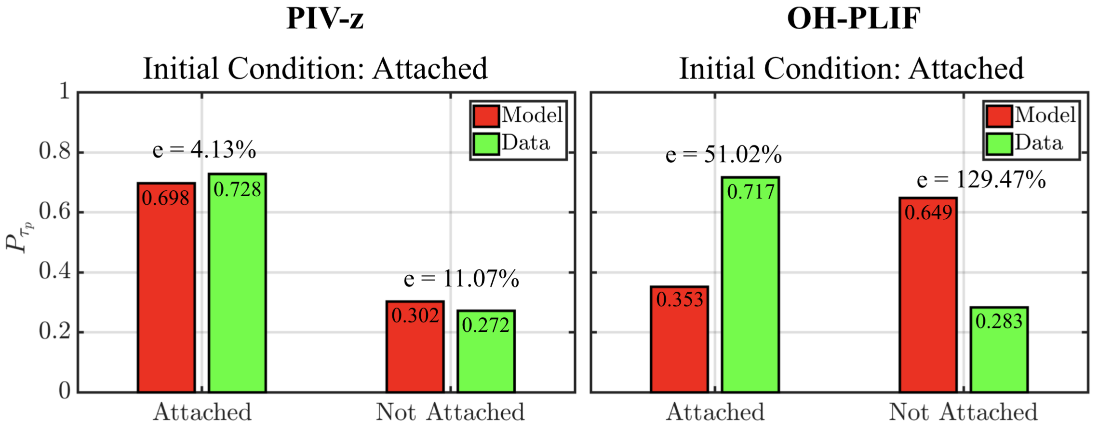


Figure 3.20: Forecast results for $\tau_p = \tau_f$ for out-of-plane velocity (left) and OH-PLIF (right) data, with the initial condition in the *attached* state (flame detachment, or liftoff). Probability values indicated in bars, and relative precent error e with respect to data-derived quantities is also shown. Y-axis is future probability at τ_p. Results shown for $\tau_p/\tau_f = 1$.

entering both transitioning and detached clusters from the attached state. As the quantity of interest in this case is the probability of *exiting* the attached state, or transitioning out of the attached state, only these two categories are considered. In Fig. 3.20, plots on the left reflect PIV-z results, and tables on the right reflect OH-PLIF results. The results show predicted probabilities (red) and observed probabilities (green) for both data types. Note that, as alluded to above, the data-derived probabilities (green bars) across PIV-z and PLIF models are nearly equal. Relative percent errors e between model and data-derived quantities are shown above the corresponding bars. Note that the PLIF model severely overshoots the probability of a snapshot leaving the attached state when compared to PIV-z (relative error of 129.47% versus 11.07%). It is clear that in forecasting flame detachment, PIV-z predictions are much more representative of the testing data than PLIF counterparts. This is in line with previously observed experimental results showing OH-PLIF as a lagging indicator of PVC-induced strain rate, a direct catalyst for flame lift-off [10].

When predicting flame attachment for $\tau_p = \tau_f$ in Fig. 3.20, errors associated with the OH-PLIF models are again expectedly higher. Note that the data-derived probabilities also match to a reasonable degree across the different models. An interesting feature to note is the increased accuracy of models across the board for prediction of flame attachment (Fig. 3.21) versus flame detachment (Fig. 3.20): both PIV-z and PLIF model relative errors observe notable drops in relative error. Nevertheless, results show that in both detachment and attachment predictions, PIV-z produces more accurate forecasts when compared to the PLIF model.

Prediction results for $\tau_p = 0.1\tau_f$ and $\tau_p = 7.0\tau_f$ are shown in Figs. 3.22 and 3.23, respectively. In each of these figures, the flame detachment predictions (similar to Fig. 3.20) are shown in the upper row and attachment (similar to Fig. 3.21) in the lower row. As in the $\tau_p = \tau_f$ predictions, the short-time predictions in Fig. 3.22 show a similar trend of higher PIV-z model performance in the detachment process. Because the forecasting time is smaller by a factor of 10, this effect is not seen to the same level as in Fig. 3.14. Despite the slightly lower level of performance here for OH-PLIF, there is high accuracy in both models in short-time predictions – they both forecast the expected state persistence with minimal relative errors. In fact, the predictions in Fig. 3.22 for PIV-z in the attachment process gives zero relative error. Note that probabilities for the ”not-attached” and ”not-detached” labels are low in Fig. 3.22 because the prediction time is extremely small.

The long-time predictions in Fig. 3.23 are provided to showcase CROM-based forecasting for timescales at or beyond model horizon times. Although trends of significantly lower OH-PLIF performance are seen here in the attachment process again, in the long-time prediction setting, both PIV-z and OH-PLIF model predictions show similar patterns. For example, they overshoot transition probability in the detachment process and undershoot in the attachment process. The model predictions (red bars) between PIV-z and PLIF are similar (especially in the detachment process) because the limit of the horizon time for both models has been reached. In other words, in the long-time limit, the models will always approximate the initial distribution of snapshots based on the training data (see Eq. 6). Additionally, if the initial distribution does not perfectly match the ergodic distribution of the Markov chains, and if there are differences between the ergodic distributions in the training and testing data, the discrepancy in the CROM predictions at times at or beyond the prediction horizon is expected to rise even higher.

For a clear illustration of prediction time effect on model performance, the variation in relative errors as a function of prediction time for the "attached" label in the detachment process is shown for PIV-z and OH-PLIF in Fig. 3.24. The red vertical lines indicate the horizon times for the respective models. As evidenced in the discussion above, the plot shows large performance advantage in the PIV-z model in the ranges of $\tau_p/\tau_f = 0.1$ to about $\tau_p/\tau_f = 5.0$, after which both errors appear to converge. Note that this apparent convergence occurs before the PIV-z horizon time and after the OH-PLIF horizon time. This importantly verifies that the prediction horizon time can indeed be used as indicator of forecasting strength when considering prediction times smaller than the horizon times. An important result from Fig. 3.24 is that although one can use horizon time as a metric for forecasting strength, the horizon time itself is not a prediction time at which one should necessarily expect accurate forecasts. Note that the OH-PLIF curve increases until a peak near its horizon time, followed by a decrease until the convergence point. In contrast, the PIV-z curve appears to increase with heavy fluctuations. The intricacies of such behaviors require further study and are out-of-scope here, though this is an object of future work.

Ultimately, with the test conditions and model inputs used, out-of-plane velocity can be comfortably classified as the most potent dataset with regards to flame transition prognostics. This essentially hints that the prediction horizon time τ_h is a good indicator to quantify the predictive strength of dataset.

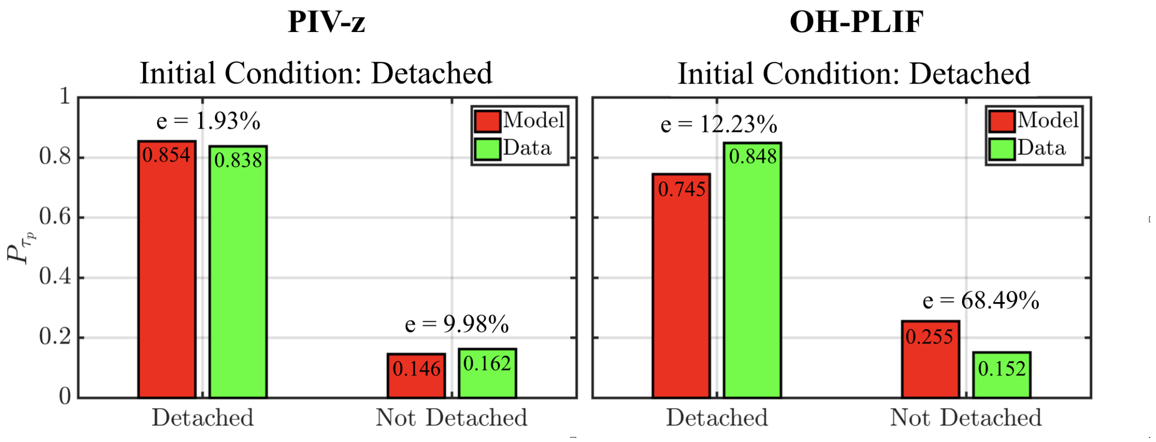


Figure 3.21: Forecast results for $\tau_p = \tau_f$ for out-of-plane velocity (left) and OH-PLIF (right) data, with the initial condition in the *detached* state (flame attachment). Probability values indicated in bars, and relative percent error e with respect to data-derived quantities is also shown. Y-axis is future probability at τ_p. Results shown for $\tau_p/\tau_f = 1$.

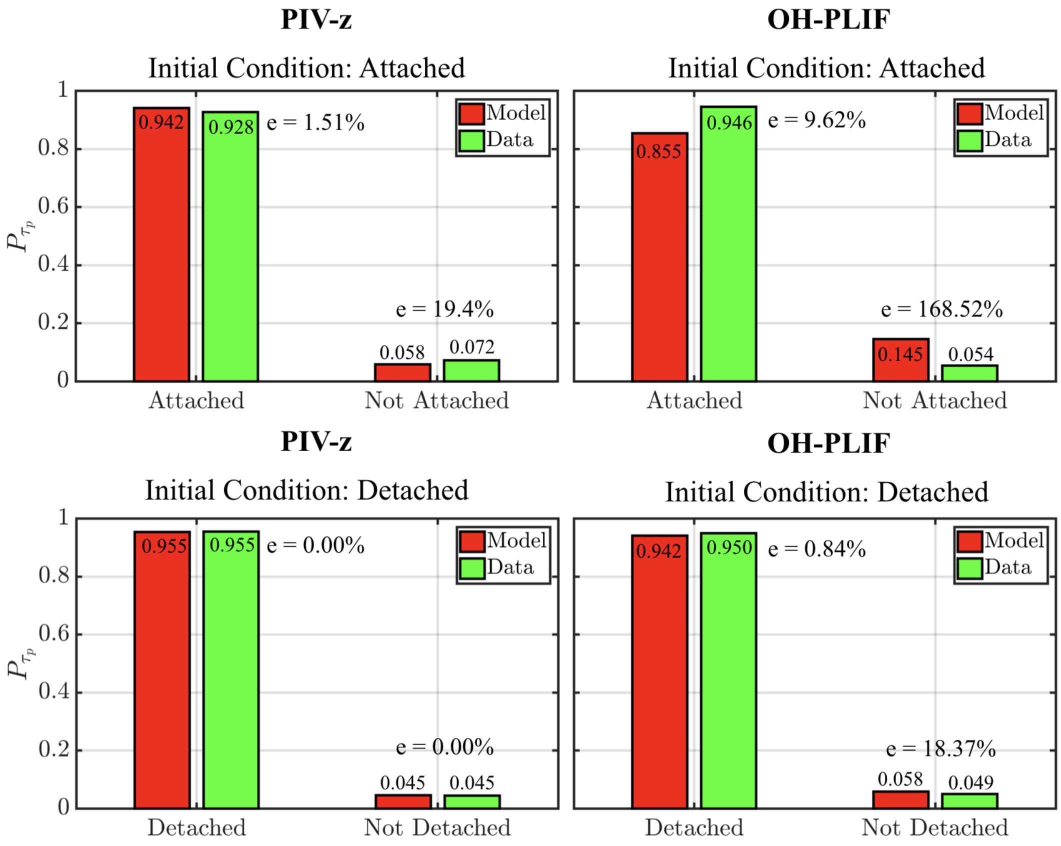


Figure 3.22: Short-time forecast results for $\tau_p = 0.1\tau_f$ for out-of-plane velocity (left) and OH-PLIF (right) data. Upper row is detachment forecast and lower row is attachment. Annotations made in same manner as Figs. 3.22 and 3.23.

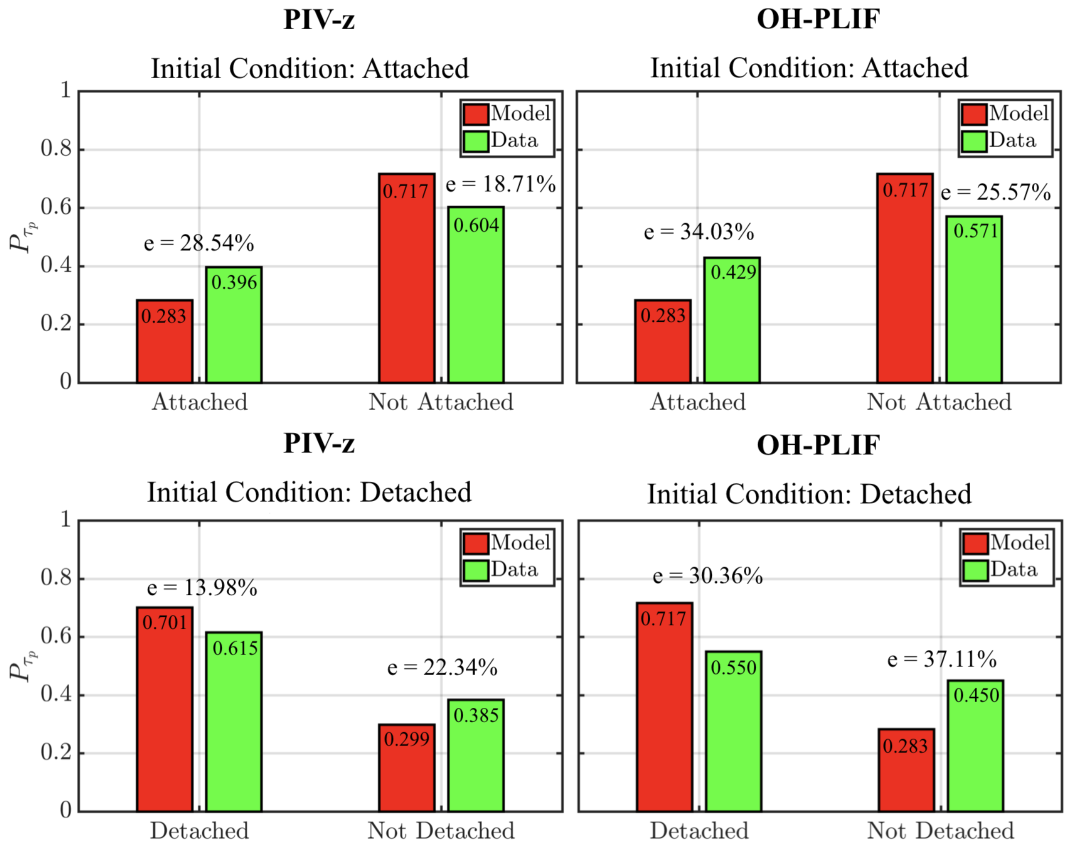


Figure 3.23: Long-time forecast results for $\tau_p = 7.0\tau_f$ for out-of-plane velocity (left) and OH-PLIF (right) data. Upper row is detachment forecast and lower row is attachment. Annotations made in same manner as Figs. 3.22 and 3.23.

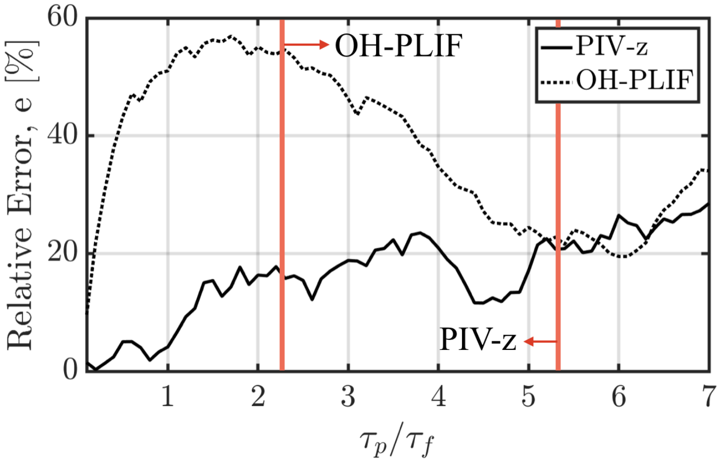


Figure 3.24: Relative error, e, as a function of normalized prediction time for the "attached" label in the detachment process. Vertical red lines indicate horizon times.

3.4.7 Summary and Conclusions

A cluster-based ROM was employed to analyze experimental data of flame transitions in a swirling combustor, and to create a model to anticipate such transitions. As an analysis tool, the method allows for the classification of data into modes that are statistically representative of the flow field and extraction of the most probable path between these modes during flame transitions. An appealing property of this process is that it can be done in an unsupervised manner, making the analysis as objective as possible. However, in order to use CROM, the number of modes must be set directly by the user. Two different approaches to set this number were presented in this work.

For the swirl combustor analyzed here, the flame detachment and attachment process were both analyzed by means of the transition matrix and modes outputted by CROM. It was shown how the modes could be associated to a macroscopic flame states (attached, detached, transition) and thereby derive a physical interpretation from the CROM output. It was found that the flame detachment process stems from an asymmetry in the flow field and for the flame. This suggests that an asymmetric process causes the flame detachment. This finding is in line with prior experimental analysis which identified PVC as the cause of flame detachment. The flame attachment appears to be triggered by a recirculation zone located at the same axial location as the flame, far from the nozzle. This recirculation seems to drive the flame down next to the nozzle. This analysis illustrates the potential of data-driven methods to analyze complex flows in a systematic way while highlighting key processes.

For forecasting purposes, the prediction horizon time of the transition matrix can be computed to help quantify the predictive power of a particular dataset. The predictive capabilities of the OH concentration and the three velocity components were compared. It was shown that z-component of velocity (swirling velocity) holds the largest prediction power while the OH signal leads to the shortest prediction window.

This is consistent with the experimental observation that the cause of the flame transitions lies in the velocity field while the flame simply responds to strain rates. The present methodology allows to confirm that the swirling velocity is a better causal indicator than the OH signal. Using the cluster labeling method, the forecasting can be formulated in terms of the macroscopic flame state rather than individual modes. The PIV-z based model significantly increased prediction accuracy while the PLIF-based model misrepresents probability considerably. The prediction horizon time is therefore a good indicator for the predictive power of a dataset. Interestingly, both PIV-z and PLIF data accurately captured the probability of attachment process. This suggests that a different physical process involving OH concentration might govern the flame attachment.

This study introduces many pathways for future work. The horizon time analysis, as it was extended to analyze forecasting strength of specific data types, can also be used to search for an optimal sensor that would hold the best predictive power. Furthermore, the forecasting tool is an important step towards constructing actuation or other control strategies in combustors. In particular, this model can used to determine acceptable probability thresholds for initiating actuation procedure, and also to understand what part of the flow to inhibit in the actuation process. While the CROM application here shows considerable promise, there is a need to fully understand how the clustering techniques can be “goal-oriented” in order to optimize the prediction for certain purposes only. The type of data is best suited for CROM forecasts was found by comparing OH-PLIF and PIV, but a similar study can also be conducted by comparing different spatial locations, proceeding with an analysis of the effect of prediction horizon time on model accuracy. These topics will be considered in future work.

3.5 Time-Axis Clustering for Modeling Turbulent Reacting Flows

3.5.1 Background

In this section, K-means clustering is used to demonstrate a new data-based reduction and decomposition technique called *time-axis clustering*, which is inspired by the idea of time series classification. Though the concept of time series classification has been around for many decades [178], its relation to feature extraction and ROM development in the context of traditional decomposition methods in fluid dynamics has not been explored. The application of time-axis clustering throughout this work is geared primarily towards the above mentioned second pathway. However, it provides a unique route towards interpretable ROM development that can facilitate connection with the first pathway. The connection between these pathways are important to recognize, as their combination illuminates the ability of ML-based techniques to construct data-derived reduced order models (ROMs) that are both a) highly accurate in a predictive sense and b) interpretable.

The idea of constructing data-derived reduced order models for both prediction and system interpretation has been explored extensively in the past via modal decomposition methods based on decoupling the spatial and temporal components of the evolving flowfield (see Sec. 2.5.1 and Ref. [330]). The spatial component, or the *mode*, exists in the same phase space as the input data and can therefore be visualized directly. In many applications, the mode is a frozen spatial structure (i.e. it is not time-evolving), which allows for the recovery of a corresponding temporal coefficient that provides a coupling to the dynamics observed in the data[3]. As described in Sec. 2.5.1, in the context of modal decomposition methods with fixed spatial modes,

[3] It is important to reiterate that modal decompositions do not necessarily have to take the form of fixed-in-time spatial modes with temporal coefficients encoding time variation – however, the general goal of all modal decomposition methods is to provide a mechanism for order reduction for physical analysis and/or simulation acceleration.

the set of scalar temporal coefficients encode the relevance or importance of the corresponding modes at a particular time instant. Then, with a limited number of such modes and their temporal coefficients, a spatial analysis on the modes combined with a temporal analysis on the joint evolution of the coefficients can be performed to make physical conclusions about the flow process contained within the scope of the data. The hope is for these physical conclusions to drive the development of more accurate and economical ROMs.

Though this fundamental notion of space-time decoupling is generally the same across many modal decomposition approaches, the manner in which this splitting is achieved can lead to vastly different data-based interpretations. A powerful yet simply implemented modal decomposition tool mentioned and applied throughout this dissertation is proper orthogonal decomposition (POD), a covariance-based method which produces the modes as orthonormal basis vectors from the data [29, 128, 185, 315]. The full set of resultant basis vectors, called the POD modes, are then truncated to construct a reduced description of the underlying system dynamics by means of preserving some arbitrary amount of the variance of the original dataset used to construct the basis. Most usefully, the basis vectors can themselves be analyzed to reveal certain qualitative aspects of the flowfield, as they represent directions in which a specific amount of flow "energy" (in the form of explained variance) is captured. As described in Sec. 2.5.2, this technique has been studied widely and applied successfully in many different fronts. Another decomposition approach that has gained considerable attention is dynamic mode decomposition (DMD) [162, 300, 339], the details of which are found in Sec. 2.5.3. In its most basic form, DMD approximates the dynamics contained in the input dataset by a single linear operator which is considered as an approximation to the Koopman transfer operator and can be obtained using a least-squares regression technique from the original time-ordered dataset. Assuming the linear operator sufficiently characterizes the dynamical complexity contained in the data, its spectral decomposition can be used to assess the underlying dynamical properties. The resulting DMD modes, which are the complex eigenvectors of linear operator, are not orthogonal, but are instead characterized by specific frequencies similar to a traditional temporal discrete Fourier transform. In other words, DMD removes the orthgonality in the global spatial modes by instead attributing single, unique frequencies to the corresponding temporal coefficients. As with POD, DMD has been effectively utilized in the fluid dynamics community due to its utility in capturing scale-separated spatiotemporal coherence [205, 301].

Despite the clear and historically demonstrated utility of the above described (and other similar) decomposition techniques, they also bring forward key issues that should be illuminated. One issue lies in the interpretability of the modes. In POD, for example, the high-energy modes are clearly interpretable, but the low-energy modes often contain extremely complex spatial structures that cannot be easily analyzed. This is especially problematic in cases where physical processes contained in the data are driven by small-scale interactions (i.e. in the case of extreme-events and ergodic intermittency) [119, 276]. A similar problem arises in DMD, where the modes are not easily interpreted for datasets that exhibit high levels of chaoticity and intermittency, since they cannot be characterized well by discrete spectra [205, 267]. Further, the ranking of DMD modes is a non-trivial task – oftentimes, a-priori knowledge of the underlying physical processes (in the form of characteristic frequencies) is required to isolate the DMD modes of interest, which on can argue is against the spirit of data-based feature extraction [19].

In light of these issues, a different take on the data-driven analysis and decomposition of complex datasets based on K-means clustering was presented in Sec. 3.4 via cluster-based reduced order modeling (CROM [140]). In CROM, K-means is applied over a set of high-dimensional snapshots, leading to the unique ability to extract intermittent and extreme events contained the data as individual clusters [18, 19, 140].

More specifically, in this type of clustering (referred to in this section as *space-axis clustering*), the high-dimensional trajectory contained in the dataset is discretized (or coarse-grained) through the K-means algorithm into a set of representative symbols, where each symbol represents a particular flow pattern or coherent shape (i.e. a mode). The evolution of the system is then represented by a string of such symbols in time, which encode the residence time of the trajectory in the different K-means clusters. This casting of the complex dynamics into a symbolic setting has led to the creation of data-driven symbolic dynamics based predictive ROMs [18, 47, 140]. Similar ideas based on symbolic dynamics (not reliant on the K-means technique) are widespread in the study of dynamical systems, and have been recently utilized to discover quasiperiodic regions in turbulence [42, 69, 284].

In this section, the aforementioned *time-axis clustering* method is presented and demonstrated, which extends the previously used *space-axis* (i.e. CROM) method used in Sec. 3.4. This is motivated by shifting the interpretation of the input dataset. More specifically, instead of interpreting the input dataset as a high dimensional multivariate time series (as is the standard fashion), it is instead interpreted as a large number of 1-dimensional time-evolutions, each given by the individual degrees-of-freedom that compose the set of snapshots. It will be shown that the modal decomposition produced by the K-means output when representing the data in this way (which can be releated to methods based on time-series classification) results in spatial modes that display discrete delineations of dynamically similar regions in physical space. As such, a unique and highly interpretable avenue for modal decomposition and data reduction is recovered.

Dataset: Before proceeding, it is emphasized that the time-axis clustering technique is applied to the same experimental dataset sourced from a premixed swirl-stabilized combustor used in Sec. 3.4, which exhibits highly complex turbulent combustion phenomena. A description of the experimental configuration is provided in

Sec. 3.4.2 and will not be repeated in this section. From the full set of 15000 available snapshots of simultaneously measured OH fields and velocity fields, a 1000-snapshot subset of the detached flame dynamics of only the OH field is used below to demonstrate the time-axis clustering technique.

The sections proceed as follows. In Sec. 3.5.2, differences between the space-axis and time-axis clustering methods are provided in the form of a general methodology. The time-axis decomposition is then produced and analyzed in Sec. 3.5.3, and closing remarks are provided in Sec. 3.5.5.

3.5.2 Methodology

Before proceeding with the time-axis decomposition output, a generalized methodology is provided. In particular, a juxtaposition of the time-axis clustering with the more commonly used space-axis framework [18, 140] applied in Sec. 3.4 is outlined. Both rely on the output of a K-means clustering algorithm [323]. The focus here is on how time-axis clustering (the output of K-means in the time-axis setting) shifts the interpretation of the representation of data reduction in phase space. An interpretation of the time-axis clustering output from the modal decomposition lens is then provided.

3.5.2.1 Space-Axis Versus Time-Axis

In usual fashion, the dataset in consideration is cast in the form of a matrix denoted $\Phi \in \mathbb{R}^{D \times N}$. Each point (or snapshot) $\phi_j \in \mathbb{R}^D$ is a column of Φ (for example, in Sec. 3.4, D is the number of grid points in a 2-dimensional flowfield), and $j = 1, \dots, N$, where N is the number of time steps (length of the time series of each degree of freedom defined by the time step Δt between each snapshot). It is standard to interpret the dataset Φ as the set of the N instantaneous D-dimensional snapshots ϕ_j, i.e. to consider Φ as a set of instantaneous high-dimensional flowfields that represent a D-dimensional time-series. Alternatively, the dataset can be interpreted as a series of D 1-dimensional coupled time-series. From this perspective, a single scalar time-series, which is a row of Φ, is denoted $w_i \in \mathbb{R}^N$ where $i = 1, \ldots, D$. Each scalar entry of w_i provides the time-evolution of the i-th grid point. We will refer to the representation of Φ by the N snapshots ϕ_j as the *space-axis* description, and the representation of Φ by the D one-dimensional time series w_i as the *time-axis* description.

In a general sense, the clustering goal is to find a set of K different groups, called *clusters*, that organize the data in such a way as to capture low within-cluster variation and high between-cluster variation. Here, the organization of the data into these K groups is done in a hard-assignment manner, which is one of the motivators for the usage of K-means. Hard-assignment means that from the K recovered groups, each data point can only be assigned to one of K groups and no other. This hard-assignment (or hard-clustering) of the data results in useful symbolic, orthogonal representations beneficial to dynamical analysis of the clustering output, and will be utilized below.

The interpretation of the data-matrix Φ directly affects how the these K groups, or clusters, are represented. For example, in the space-axis setting, the dataset is represented as a series of N instantaneous flowfields $\phi_j \in \mathbb{R}^D$. As such, the clusters partition the N instantaneous snapshots into K clusters. In contrast, in the time-axis setting, the dataset is represented as a series of D one-dimensional time evolutions $w_i \in \mathbb{R}^N$. Then, the resulting K clusters instead bin the D degrees-of-freedom and not the N snapshots. A single member of one of K clusters in the time-axis setting is the entire time-evolution of one of D grid points. As such, the time-axis clustering can be thought of as a type of time-series classification of the grid points.

Depending on whether the time- or space-axis representation of ϕ is considered, to produce the K clusters, the K-means algorithm discretizes either the $\mathbb{R}^N$ or $\mathbb{R}^D$

phase space respectively in a coarse-graining procedure. As described in Sec. 3.2, the resulting clusters are guaranteed to be non-overlapping subspaces of the respective phase space (a member of one cluster is unique to that cluster), and are organized in the set $\mathcal{K} = \{\mathbb{K}_1, \mathbb{K}_2, ..., \mathbb{K}_K\}$, where each element of $\mathcal{K}$ is a single cluster. Note that the set $\mathcal{K}$ is fundamentally different depending on which representation (space- or time-axis) is considered. In particular, the space-filling quality in $\mathbb{R}^D$ (space-axis) or $\mathbb{R}^N$ (time-axis) effectively translates to the following: a) in the space-axis interpretation of Φ, any conceivable flowfield defined in $\mathbb{R}^P$ can be assigned to one of K clusters, and b) in the time-axis interpretation, any conceivable univariate time-series consisting of N time-steps defined in $\mathbb{R}^N$ can be assigned to one of K clusters. The assignment process itself is explained further below.

More generally, if the value S is equal to either D (space-axis) or N (time-axis), the set $\mathcal{K}$ must observe the following: 1) each $\mathbb{K}_i \subseteq \mathbb{R}^S$, 2) the $\mathbb{K}_i$ are strictly non-intersecting in the S-dimensional phase space, 3) the union of all elements in $\mathcal{K}$ exactly reconstructs $\mathbb{R}^S$, and 4) the cardinality of $\mathcal{K}$, which is the number of clusters K, must strictly lie between 1 and S.

Through the variance-minimization procedure outlined in Alg. 1, the K-means algorithm outputs the non-intersecting $\mathbb{K}_j$ clusters based on the placement and convergence of the centroids denoted by the set $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$, where $\mathbf{c}_i \in \mathbb{R}^S$ ($S = P$ in space-axis case and $S = N$ in time-axis). Recall that centroids are computed as regional averages of the data in the corresponding clusters. For example, in the time-axis case ($S = N$),

$$c_j = \frac{\sum_{i=1}^{D} \mathbb{1}_{\mathbb{K}_j}(w_i) w_i}{\sum_{i=1}^{D} \mathbb{1}_{\mathbb{K}_j}(w_i)}. \tag{3.10}$$

Here, $\mathbb{1}_{\mathbb{K}_j}$ is the indicator function for each cluster $\mathbb{K}_j$, where

$$\mathbb{1}_{\mathbb{K}_j}(w_i) = \begin{cases} 1, & \text{if } w_i \in \mathbb{K}_j \\ 0, & \text{otherwise.} \end{cases} \tag{3.11}$$

Note that the above definition of the centroid is equivalent to that provided in Eq. 3.3 – the version above based on the indicator is used here for notational convenience. For classification purposes, the time-series w_i is said to reside in cluster $\mathbb{K}_j$ if the snapshot is closest to centroid c_j in the N-dimensional Euclidean sense. In other words, if $\mathbb{1}_{\mathbb{K}_j}(w_i) = 1$, then the i-th member is said to be *represented by* the j-th centroid. The above equations are analogous for the $S = D$ space-axis case, where instead the indicator functions act on the D-dimensional snapshots ϕ_i.

This implies that depending on the clustering dimension, the centroids c_j are either 1) images made of D pixels that represent the associated clusters $\mathbb{K}_j$ (space-axis) or 2) time-evolving quantities that represent the evolution of dynamically similar degrees-of-freedom (time-axis). Ideally, each centroid should be as close as possible to its assigned members such that the centroid represents the behavior of the entire cluster well. Figure 3.25 shows illustrative examples of a single cluster using both space-axis and time-axis frameworks. The comparison shows that small temporal fluctuations are damped by the centroid in the time-axis setting in the same way small spatial fluctuations are damped by the centroid in the space-axis clustering. The level at which this damping occurs is related directly to the parameter K.

3.5.2.2 Decomposition Perspective

The clustering procedure results in two primary outputs: a set of centroids $\mathcal{C}$ and labels provided by the indicator functions. The labels are the cluster assignments of either a) the N instantaneous snapshots in the space-axis setting, or b) the time-

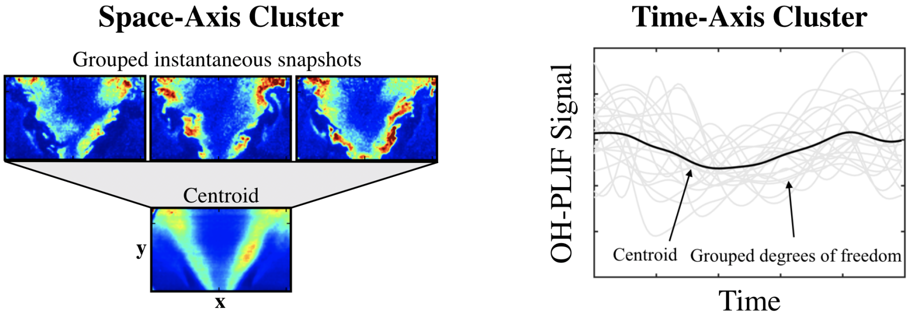


Figure 3.25: **(Left)** Representation of a space-axis cluster used in Sec. 3.4. **(Right)** Representation of a time-axis cluster using the approach described in Sec. 3.5.2. Clustering outputs generated from planar OH PLIF dataset sourced from model DLR combustor configuration described in Sec. 3.4.2.

evolution of D grid points in the time-axis setting. The information contained in the centroids and labels can be organized in a way that allows the clustering output to be interpreted as a traditional modal decomposition. Casting the clustering in the decomposition framework provides useful insights, as will be seen below.

As described in Chapter II, Sec. 2.5.1, modal decomposition methods with the goal of space-time decoupling split the data Φ into 1) a finite set of spatially frozen modes, and 2) time-evolving scalar weights for each mode called temporal coefficients. The modes are stationary and exist in $\mathbb{R}^D$; they can be interpreted as time-averaged spatial directions which can be visualized. Formally, for a finite set of modes, the decomposition approximates the snapshot ϕ_i as

$$\widehat{\phi}_i = \sum_{k=1}^{K} \mathbf{T}_i^k r_k, \tag{3.12}$$

where $\widehat{\phi}_i \approx \phi_i$ ($\widehat{\phi}_i$ recovers the mean flow when $K = 1$ and the exact field when $K = S$). Here, the number of modes K corresponds directly to the number of clusters. The quantity r_k the k-th mode. All K modes can be contained in the matrix $\mathbf{R} \in \mathbb{R}^{D\times K}$, where r_k is the k-th column of $\mathbf{R}$. The quantity $\mathbf{T}_i^k$ is a scalar corresponding to the temporal coefficient of snapshot i for mode k. Similarly, if all temporal coefficients for all modes are contained in the matrix $\mathbf{T} \in \mathbb{R}^{K\times N}$, then $\mathbf{T}_i^k$

indexes the i-th column and k-th row of $\mathbf{T}$. The decomposition provided in Eq. 3.12 is a convenient way to relate the space- and time-axis clustering outputs. First, the space-axis perspective will be considered followed by the time-axis perspective.

In space-axis clustering used in the CROM approach of Sec. 3.4, the centroids are conditional averages of the snapshots and are stationary flowfields. Since the centroids c_k are D-dimensional and the clustering occurs over the various ϕ_i (a D-dimensional time series), classification using the indicator functions (Eq. 3.11) produce a single cluster label/assignment for each snapshot. To recover the temporal coefficients, the cluster assignments can be organized in the columns of the matrix $\mathbf{T}$. The i-th column in $\mathbf{T}$ represents the cluster label for the i-th snapshot through the indicator function: its j-th element is given by $\mathbb{1}_{\mathbb{K}_j}(\phi_i)$, $i = 1, \ldots, N$. If the centroids are organized in the columns of $\mathbf{R}$ (i.e. if $r_k = c_k$), then, in the space-axis setting, the temporal coefficients are the cluster labels for each snapshot and the spatial modes are the centroids. Due to the nature of the indicator function, the rows of $\mathbf{T}$ can be scaled to be orthogonal in the space-axis decomposition such that

$$\langle \mathbf{T}^i, \mathbf{T}^j \rangle = \delta_{ij}, \tag{3.13}$$

where $\mathbf{T}^i$ and $\mathbf{T}^j$ represent the i-th and j-th rows of the coefficient matrix $\mathbf{T}$, respectively.

In time-axis clustering, the centroids become conditional averages of a set of grid point time-series. As such, the centroids themselves are time-evolving quantities, as shown in Fig. 3.25. The centroids are interpreted as the temporal coefficients: each row of $\mathbf{T}$ can be populated with the K time-axis centroids c_k. What remains is the determination of the time-axis modes. Just as the space-axis temporal coefficients were obtained through the *snapshot* labels, the time-axis modes are obtained through the *grid-point* labels. In the time-axis decomposition, each D-dimensional column of

$\mathbf{R}$ contains the k-th time-axis mode. This is obtained through the indicator function for the time-series of each of the D grid points: for mode k, the p-th element is obtained as $\mathbb{1}_{\mathbb{K}_k}(w_p)$, where $p = 1, \ldots, D$ indexes an individual grid point or cell in the domain. The properties of the indicator functions for each cluster provides (with scaling assumed) orthogonal modes such that

$$\langle r_i, r_j \rangle = \delta_{ij}. \tag{3.14}$$

It will be shown in Sec. 3.5.3 how this mode orthogonality provided by the time-axis decomposition is especially beneficial for interpretation and feature extraction.

Since the interpretation of the data matrix is flipped in the space-axis setting, in space-axis clustering, the coefficients are orthogonal and the modes (which are the centroids) are not. In time-axis clustering, the reverse is seen: the modes are given by the cluster labels and are orthogonal, and the temporal coefficients are the centroids, which are not orthogonal. Note that in Eq. 3.12, it has been assumed that information has been lost in the decomposition – this is a product of the coarse-graining induced by the clustering, which assumes $K \ll S$. Only in the case of $K = S$ does $\widehat{\phi}_i = \phi_i$, though this renders an effectively redundant clustering output. Further, it has been assumed that each mode in Eq. 3.12 has been appropriately scaled by the number of members in the corresponding cluster.

3.5.3 Time-Axis Decomposition

Here, the time-axis decomposition as applied to the OH PLIF detached flame dataset is investigated. A demonstration is provided with $K = 5$. Then, the effect of changing to a higher value is of $K = 7$ is shown. Note that although these values for K serve the purpose of demonstrating the technique, details of quantitative methods related to choosing optimal K values are not discussed here.

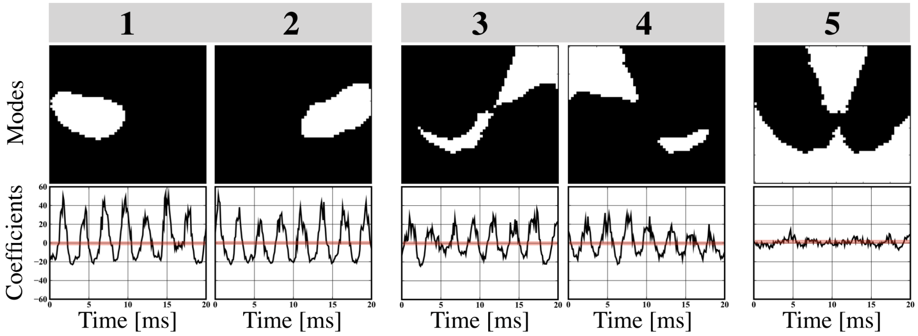


Figure 3.26: (Top row) Time-axis modes (columns of $\mathbf{R}$) for $K = 5$ clusters. (Bottom row) Temporal coefficients (centroids) for each mode, with red line indicating $y = 0$ mark.

3.5.4 Demonstration of the Method

The time-axis clustering results for a the OH PLIF detached flame are shown in Fig. 3.26. Shown are the individual modes separately for $K = 5$. The orthogonality of the modes is clearly evident since the regions are non-overlapping in the physical space (this is a product of hard-assignment of pixels, or alternatively, enforcing non-intersecting clusters in $\mathbb{R}^N$). The non-zero values in these modes (white regions) represent pixels which exhibit similar behavior in the time-span given by the N snapshots in the data. The centroids, which are the temporal coefficients contained in T, are also shown for each corresponding delineated region in the lower row of Fig. 3.26. Due to the non-overlapping property of the clusters, all of the modes can be visualized at once, as seen in Fig. 3.27. This creates a *segmented flowfield* based on the time-axis clustering of the scalar field in Φ, where the different colors represent the 5 different spatial modes shown in Fig. 3.26.

A number of useful qualities can be extracted from the decomposition. Firstly, the clustering algorithm appears to separate the non-reacting region (cluster 5) from the reacting region (other 4 clusters). In other words, the non-zero values for mode 5 correspond to regions in which little-to-no OH signal fluctuation is present. This is evidenced in the corresponding centroid, which is nearly zero at all times. The time-axis decomposition then uses the remaining 4 clusters to partition the physical domain corresponding to region in which there is OH signal fluctuation present in the time-series. It can clearly be seen that, among these 4 modes, the time-axis clustering results recover large amounts of pair-wise symmetry for this particular dataset. These pairings are provided in Fig. 3.26. A crucial detail is that the symmetry is captured in cluster structure, but not in cluster identification. For example, the cluster groupings on the left side of the domain are similar in structure to those on the right, but the cluster assignments (labels) for these pixels are different. This is more clearly visualized in Fig. 3.27.

Furthermore, when examining the modes, many of the spatial structures show neighborhoods of pixel groupings; in other words, pixels nearby one another tend to belong to the same time-axis cluster, which points to the fact that dynamical similarity can be captured in discretely defined flow features which may be joint or disjoint in the physical space. For example, modes 3 and 4 (second pair of symmetric clusters in Fig. 3.26) are unique from the rest because they exhibit disjoint flow features. In particular, they identify a relation between the flame anchoring point just above the burner exit and the OH signal located in the top of the domain on the opposing side, where rapid turbulent fluctuations dominate. The decomposition therefore extracts much of the relevant flame-turbulence interaction generated by the swirl-stabilized flame dynamics (from the perspective of OH) in this single mode pair; the orthogonality of the modes allow for these interactions to be clearly visualized through the pixel groupings. The other mode pair (modes 1 and 2) show the characteristic symmetric trends, but is localized to the middle y-axis region of the domain above the flame anchor, and does not contain disjoint groupings of pixels.

More information can be extracted when pairing the spatial information contained in the modes to the corresponding time-evolving centroids, which are the temporal coefficients in the modal decomposition context. Within each mode-pair, the centroids are similar: the only major differences are phase shifts. Across the different mode pairs, centroid behavior is different: for example, cluster 5 displays minimal centroid fluctuation, indicating that this particular region of the domain can be almost perfectly characterized by the mean OH signal, and is therefore not interesting from a dynamical standpoint. The centroids between cluster pairs 1/2 and 3/4 are different in that they capture different peak amplitudes at similar frequency. In particular, modes 1 and 2 reflect higher peak amplitudes when compared to modes 3 and 4, which implies that the first mode pair better localizes the effect of the precessing vortex core (PVC) structure, and the second mode pair (3 and 4) reflects localized regions in which entrainment effects imposed by the PVC dynamics (secondary PVC effects) are more prominent.

Since the spatial modes are discrete and non-overlapping, they may be interpreted more easily than, for example, POD modes which are also orthogonal. This is because a single pixel is attributed to only one trend (given by the corresponding centroid) and no other. Further, unlike POD which may contain uninterpretable, complex flow features in the low-energy modes, the time-axis modes capturing low fluctuations (such as cluster 5) can be easily interpretable. The trade-off is the dependency on K. Though the above interpretations provided by the time-axis decompositon are powerful, there is no reason to believe that the choice of $K = 5$ is the most optimal from an analysis standpoint. This choice of K was motivated by a user-decision, in that it was low enough to simplify analysis but high enough to capture physically relevant regions. One can argue that the above conclusions derived from the $K = 5$ selection are arbitrary as a result. As such, the effect of changing K must be assessed. This is shown in the right plot of Fig. 3.27, which shows the segmented flowfield for $K = 7$. It can be seen that with an increase in K, the domain classified as the steady region is for the most part unchanged; instead, the decomposition chooses to refine the more relevant reacting regions into more clusters while maintaining the

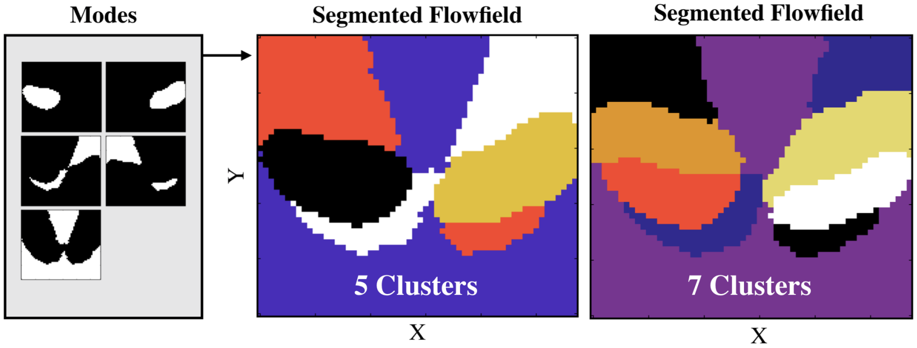


Figure 3.27: Examples of segmented flowfields obtained from the time-axis decomposition using $K = 5$ and $K = 7$. Note that cluster colors are not consistently defined across the two images.

symmetric cluster structure. This structural consistency with increasing K (which was also observed for even higher K values) gives a form of loose validation for the $K = 5$ analysis conducted above.

3.5.5 Summary and Conclusions

In this section, the idea of time-axis clustering as a new decomposition method using the K-means algorithm was outlined and demonstrated at a basic level in the context of on an experimental swirl-stabilized combustor dataset. Since the spatial time-axis modes are discrete and non-overlapping, they provide a much more clear route to interpretation than other decompositions. This is because a single pixel is attributed to only one trend (given by the corresponding centroid) and no other. The consideration of the flow features contained in these discrete time-axis modes (which are the cluster assignments derived from the time-evolving centroids) led to the recovery of discrete, localized identification of dynamically inactive and active regions in space. Further, the ability to isolate a single dynamical pattern (the time-axis centroid) to the time evolution of one pixel via the time-axis modes provides a clear visualization of dominant spatial lengthscales and correlation as a function of the unique dynamical patterns represented by the corresponding centroids.

From the demonstrations shown in the results above, the time-axis clustering framework provides a promising avenue for a different type of decomposition that emphasizes the spatial localization of dynamically similar degrees-of-freedom. As such, the finer and more subtle implications of the time-axis decomposition framework from both the reduced order modeling perspective and K-selection are currently being pursued, and will be explored in future work. It is noted that the idea of producing segmented fields amenable to localized model development is also used in the main contribution of this dissertation by means of feature space clustering (running K-means in the space of mass fractions, for example) – this is covered in detail in Chapters IV and V.

3.6 Conclusion

The main goal of this chapter was to introduce some fundamental concepts of K-means clustering. In summary, in Sec. 3.2, the underlying objective function (the WCSS) was described and an iterative algorithm by which the objective function can be minimized (Alg. 1) was presented. Additional properties and nuances related to the K-means clustering output, which is a set of centroids that serve as generators for an underlying Voronoi tessellation, were then outlined in Sec. 3.3 – included for illustrative reasons were comparisons with spectral clustering and POD. Overall, emphasis was placed on interpretation of the K-means clusters from the fluid dynamics and modeling perspective.

Given the background of the standard K-means approach presented in Sec. 3.2 and 3.3, the remainder of the chapter was dedicated to applications of the K-means algorithm for analysis, modal decomposition, and reduced-order modeling of turbulent combustion phenomena. In Sec. 3.4, using K-means clustering on a dataset consisting of time-resolved planar simultaneous experimental measurements of OH and velocity fields, a cluster-based ROM was employed to analyze experimental data of flame transitions in a swirling combustor, and to create a model to anticipate such transitions.

As an analysis tool, the intepretability garnered by the K-means centroids allowed for (a) the classification of data into clusters representing attached, detached, and transition regions of phase space that are statistically representative of the flow field data, and (b) extraction of the most probable path between these clusters during the flame transition phenomena. In Sec. 3.5, the idea of time-axis clustering as a new modal decomposition method derived from a different extension of the K-means algorithm was outlined and demonstrated at a basic level using the same experimental dataset. In particular, the time-axis approach showed how a re-interpretation of the input data matrix allows K-means to produce spatial modes that are discrete and non-overlapping, providing a potentially clearer route for physical interpretation and decoupling of complex reacting flow processes than other decomposition approaches.

Note that the applications presented in Sec. 3.4 and 3.5 were included in this chapter to provide the reader some additional physical context regarding the robustness and versatility of the K-means algorithm in modeling turbulent reacting flow. As such, these sections are precursors to the classification-based regression approach discussed next in Chapters IV and V, which constitutes the main research contribution of this dissertation (see Sec. 2.6).

CHAPTER IV

Data-driven Classification and Modeling of Combustion Regimes in Detonation Waves

4.1 Introduction

In turbulence-driven combustion processes, computational modeling of complex chemical reactions pose a perpetual challenge [276]. This challenge, as discussed in Chapter I in the discussion surrounding Fig. 1.7, is primarily due to the computational burden associated with solving highly nonlinear stiff equations for the advancement of chemical state, which becomes especially prohibitive in combustion environments that require detailed chemical mechanisms to properly describe the effect of turbulence-chemistry interactions at small length and time scales. Thus, in a general sense, the primary goal of source-term modeling is to reduce the cost of these computations through the construction of physics-inspired reduced representations using data-driven techniques or other means. This objective has ultimately led to the development of a wide variety of modeling approaches related to the underlying goal of computationally-efficient source term estimation [258, 276]. Common chemical kinetics modeling pathways to accelerate source term evaluations grounded in physics include the class of flamelet/progress variable approaches [258], in-situ tabulation approaches [26, 263], and virtual chemistry approaches [46, 90] – for greater detail, the reader is directed to the physics-based combustion modeling survey provided in Chapter 2.4.2 and the references therein.

As described in Sec. 2.4.7, the disadvantage in the above physics-based modeling strategies is lack model extendability – the validity of the model is closely linked to the selection of canonical systems. In other words, physics-based models by design are derived from simple canonical configurations, and therefore cannot be extended reliably to complex, full-geometry simulations that contain a wide variety of flow turbulent combustion regimes. In light of these shortcomings, an alternate data-based route for kinetics modeling that must also be mentioned is enabled by artificial neural networks (ANNs), which was covered in detail in Sec. 2.5.6. ANNs are well-suited for source term modeling and build upon the above techniques for many reasons: 1) they utilize nonlinear representations of linear combinations of the thermochemical state to generate source term predictions, 2) they can be more memory-efficient than traditional tabulation-based approaches, and 3) they are well-primed for the newer GPU-accelerated HPC architectures. ANNs have been used both recently and in previous decades for combustion modeling for these above reasons [60, 145, 306].

While each of the above techniques have been used for different combustion systems, the focus of this dissertation as described in Chapter I is on emerging hypersonic propulsion concepts such as RDEs and scramjets. A unique need in the modeling of such combustors is the high-fidelity simulation of long-time behavior for design purposes – see Fig. 1.4 and the associated discussion. Currently, the computational cost associated with chemistry limits the total simulation time to tens of milliseconds, which is not sufficient to ensure that the system has reached a statistically stationary state [295]. The focus of this chapter is on simulation acceleration for RDEs; more specifically, the goal is to develop a machine learning-based chemistry representation that vastly accelerates the integration of chemical source terms to enable long-time RDE simulations.

The primary physical characteristics of RDE device operation as well as non-idealities are repeated here from Sec. 1.2 for convenience. In a typical RDE as shown in Fig. 4.1, a detonation wave propagates azimuthally in an annular chamber, processing a mixture of fuel and air injected axially (or radially, depending on the configuration) from the bottom of the device. However, the combustion processes within an RDE can be highly unsteady and chaotic, with both detonative and deflagrative combustion due to mixture non-uniformity and adverse wave behavior. The detonation process is highly non-ideal, with a reaction zone significantly broader than the theoretical expectation, flanked by parasitic deflagration ahead of the wave and slow, delayed heat release that extends far behind the wave and homogenizes the mixture [53, 272]. Since detonations occur over short length and time scales, the computational grids used to simulate the geometries need to fully resolve the chemical reactions and flow gradients. Further, the stability of the detonation wave is highly influenced by the fuel-air mixing process which is driven by the high-shear generated from the injection scheme. Hence, capturing the turbulent mixing mechanisms and their interplay within the unsteady detonation process is crucial, but expensive. From the modeling side, challenges emerge from the fact that the detonation domain is segmented into distinct detonation, deflagration, and transitional (or intermediary) regions [273]. Thus, it is crucial from both perspectives to systematically differentiate between the different forms of combustion. These combustion regimes vary spatially, resulting in highly stiff chemistry contained within small portions of the domain that greatly increase the computational cost of these studies. Therefore, to enable the long-time simulations of RDEs, associated models should both a) take into account differences between these complex spatially-varying regimes and b) properly leverage the rich datasets generated by the well-resolved numerical simulations.

The contribution of this chapter lies in the demonstration of the classification-based regression strategy for source term estimation described in Sec. 2.6 and Fig. 2.18,

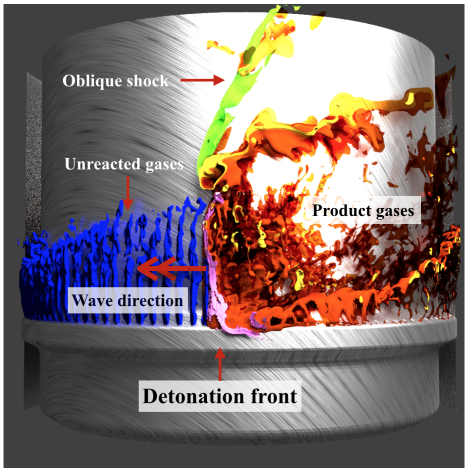


Figure 4.1: Full-scale simulation snapshot with the detailed flow features of a typical RDE combustor [295].

which constitutes a different data-driven combustion modeling approach that addresses the above mentioned numerical and modeling challenges. The approach consists of two steps. In the first step (classification), a standard K-means clustering algorithm is used in the thermochemical composition space to extract the varying regimes of combustion in the detonation wave structure in an unsupervised fashion. In the second step (regression), the clustering output (which produces a delineated flowfield) is used to drive the training of several ANNs, each of which produces source term estimations that are localized to the regimes identified in the first step by design. More specifically, using datasets generated from direct numerical simulations of canonical detonating flows relevant to the study of RDE physics, the primary focus of this work is to show that a) the clustering output recovers physically relevant delineations of combustion regimes in the detonation wave structure in an unsupervised manner, and b) the source term estimations obtained from ANNs tailored to these different combustion regimes (i.e. local ANNs) are more accurate than the estimations obtained when *not* considering the clustering output during ANN training (i.e. global ANN).

The paper proceeds as follows. In Sec. 4.2, the sample detonation datasets derived from past direct numerical simulations are described, and the reasoning for their selection is explained. In Sec. 4.3, the clustering results are discussed in terms of detonation wave structure delineations termed segmented fields. In Sec. 4.4, the localized training of ANNs derived from the segmenteed fields produced in Sec. 4.3 is carried out. Both improvements and advantages of the classification-based regression approach are discussed. Lastly, in Sec. 4.5, concluding remarks and directions for future work are presented.

4.2 Description of Data

A direct numerical simulation (DNS) approach is used to generate the models for the detonation wave structure. In this section, the numerical details of the associated solver are described and the details of the training/testing datasets used for the data-driven modeling results throughout the paper are presented.

4.2.1 Numerical Solver and Chemical Mechanism

The numerical simulation must capture the small-scale turbulence structures and the induction region, which is very thin compared to other length scales that dictate detonation wave behavior. For most chemical compositions and flow conditions, the detonation wave – a primary shock wave coupled with a reaction zone – is on the order of a few micrometers in width. The flow gradients must therefore be accurately resolved to capture the interactions between the fuel-air mixture and the detonation wave. Furthermore, detailed chemical mechanisms are required to model the species transient behavior across the wave accurately. To this end, the governing equations for fluid flow consist of the mass, momentum, and energy conservation equations augmented by the species conservation equations that incorporate chemical reactions. The system of equations is closed using the ideal gas equation of state.

The Navier-Stokes equations in compressible form as described in Sec. 2.2 are incorporated in the in-house compressible flow solver UTCOMP, which has been validated for a range of shock-containing flows [87–89]. The detailed chemical kinetics for hydrogen-oxygen combustion with a nitrogen diluter is derived from the 9-species 19-reaction chemical mechanism of *Mueller et al.* [212] using CHEMKIN-based subroutines [144]. The solver has been validated for hydrogen-air and hydrogen-oxygen detonation using a series of one-, two-, and three-dimensional canonical cases [273]. A structured grid is utilized with a cell-centered, collocated variable arrangement. A 5th order weighted essentially non-oscillatory (WENO) scheme [139] is used for computing the non-linear convective fluxes and the non-linear scalar terms are calculated using a quadratic upstream interpolation for convective kinematics (QUICK) scheme [126]. A 4th order central scheme is used to calculate the diffusive terms and a 4th order Runge-Kutta scheme is used for temporal discretization. Additional details on solver parameters used in this work are provided in Ref. [273].

4.2.2 Training and Testing Data

In the investigation of RDE flow physics, the primary flow features stem from the injector dynamics and the mixture inhomogeneity within the annulus. The fuel-air mixture is highly stratified due to the turbulent mixing characterized by recirculation zones and free-shear/jet interactions within the combustor. These interactions can be simulated through canonical channel flow geometries. The training dataset is derived from case 1b of the linear injector array simulation of Ref. [273]. The testing dataset is sourced from a channel detonation simulation with a stratified fuel-air mixture, as implemented in Ref. [272]. The schematic of the numerical configuration and fuel-air distribution in the training and testing datasets are given in Fig. 4.2. The two cases are at similar operating conditions and the same combustion regimes are present, allowing for a convenient way to assess confidence in the data-driven approach.

The full *training dataset* is a set of snapshots from a linear injector array, known

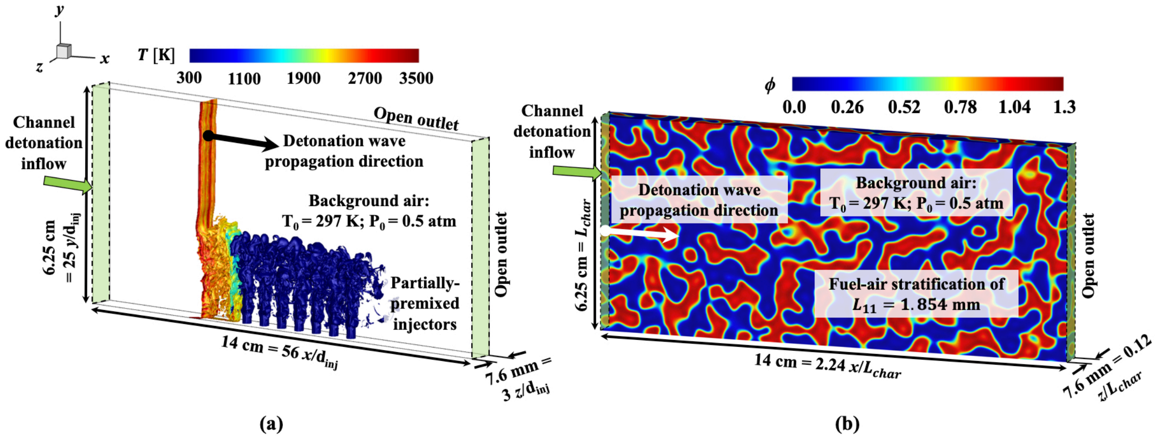

y
z
x
T [K]
300 1100 1900 2700 3500
Open outlet
Channel detonation inflow
Detonation wave propagation direction
Background air: T_0 = 297 K; P_0 = 0.5 atm
6.25 cm = 25 y/d_{inj}
Partially-premixed injectors
Open outlet
14 cm = 56 x/d_{inj}
7.6 mm = 3 z/d_{inj}
(a)
ϕ
0.0 0.26 0.52 0.78 1.04 1.3
Channel detonation inflow
Detonation wave propagation direction
Background air: T_0 = 297 K; P_0 = 0.5 atm
Fuel-air stratification of L_{11} = 1.854 mm
6.25 cm = L_{char}
Open outlet
14 cm = 2.24 x/L_{char}
7.6 mm = 0.12 z/L_{char}
(b)

Figure 4.2: Configurations of the (a) linear injector array with isocontour of H_2 = 0.016 and OH = 0.0053 colored by temperature and (b) channel with stratified fuel-air mixture of integral length scale L_{11} = 1.854 mm, which serve as the training and testing datasets, respectively.

as the linearized model detonation engine (LMDE), as depicted in Fig. 4.2a. In the LMDE configuration, a fully-developed detonation wave processes a partially-premixed stoichiometric hydrogen-air mixture injected from an array of 15 injectors. Here, the jet turbulent mixing controls the local gas composition before the detonation wave processes the mixture. For numerical cost considerations, the operating pressure was lowered to 0.5 atm with an ambient temperature of 297 K when simulating the LMDE. Upon running the respective simulations at these prescribed operating conditions, a set of thirteen consecutive two-dimensional snapshots, separated by $dt = 0.25$ μs, was extracted along the depthwise mid-plane of the domain. The thirteen snapshots capture the detonation wave structure as it passes through the 12th injector.

The full *testing dataset* is composed of ten snapshots ($dt = 0.25$ μs) sourced from a canonical simulation of channel detonation with a quiescent stratified fuel-air mixture. The range of equivalence ratios is constrained from 0 to 1.3. This is shown in Fig. 4.2b. In a confined channel of dimensions identical to the LMDE, fuel-air stratification is introduced through a method used for turbulent mixing studies in homogeneous isotropic turbulence; further details are provided in Ref. [272]. Similar to the training

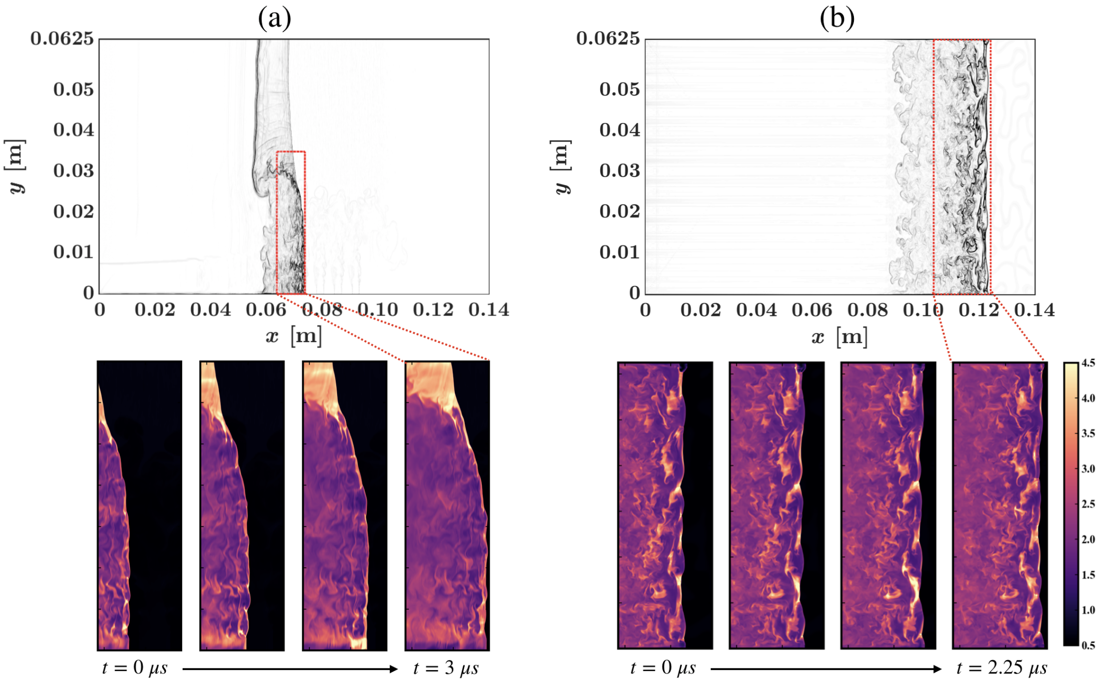


Figure 4.3: The top row shows numerical Schlieren images of the detonation wave through (a) the linear injector array and (b) a stratified fuel-air mixture in a confined channel. The bottom row shows a few of the training and testing set snapshots displaying the time evolution of density (units of kg/m^3). For convenience, the first snapshot in the dataset sequence is denoted as the initial state ($t = 0$).

dataset, the ambient pressure and temperature are 0.5 atm and 297 K. The snapshots are recorded as the detonation wave reaches near the end of the channel. Though the testing set data may seem less physically complex, the gas mixture spans a greater range of thermochemical state space in a more controlled manner in comparison to the training dataset, allowing for ideal model testing conditions. Further, this testing set particularly useful as it contains similar detonation-induced combustion regimes as the LMDE training case. However, the differences in spatial fuel-air distribution and influence of each regime on wave propagation due to the presence of fuel stratification make it a good candidate for assessing model extrapolation.

As a visual example, numerical Schlieren images of the depthwise mid-plane of a single snapshot (one of thirteen) of the training (left) and testing (right) datasets are shown in Fig. 4.3a. The datasets are cropped to the outlined regions which capture all of the combustion regimes around a detonation wave: a) an ambient fuel-air mixture, b) a shock-separated region in the absence of fuel, c) a primary detonation region, and d) a post-detonation deflagrative combustion region trailing the shock front. After cropping, each training snapshot is composed of 1011 pixels in the x direction and 363 in the y direction, resulting in 3.7×10^5 grid points per snapshot. Each testing snapshot consists of 411 pixels in the x direction and 858 pixels in the y direction, for a total of 3.5×10^5 grid points per snapshot.

As a final note, in the regime classification analysis (Sec. 4.3), all 13 of the LMDE snapshots are used in training and all 10 of the stratified mixture snapshots are used in testing. To demonstrate the source-term estimation concept in Sec. 4.4, only one of 13 available LMDE snapshots is used in training (i.e. data from one snapshot is used to train the neural networks), and one of 10 available stratified mixture snapshots in testing. Additionally, for better assessment of the model dependence on configuration, the source term estimation ANN models will also be evaluated on an unseen snapshot from the same LMDE training configuration at a future time.

4.3 Regime Classification

The goal in this section is to a) classify grid points within the domain to macroscopic regions of interest, and b) assess the physical significance of these classifications. In particular, the demarcations of regions associated with detonation and deflagration are desired, as they will guide a localized modeling procedure in Sec. 4.4. Although there are many potential routes available to the user for successful labeling, the one used here is the K-means clustering approach described in Chapter III. For algorithmic details and information on the formulation of the standard K-means algorithm, the reader is referred Sec. 3.2. The finer mathematical details of K-means can be found in Refs. [13, 323].

4.3.1 K-means Clustering Context

Consider a set of grid points $\Phi = \{\phi_1, \phi_2, ..., \phi_N\}$ extracted from one or more numerical or experimental snapshots. For example, in the training dataset described in Sec. 4.2, there are 13 snapshots each composed of 3.7×10^5 grid points, yielding a total number of grid points $N = (3.7 \times 10^5) \times 13$. Further, consider another set $\mathcal{F} = \{F_1, F_2, ..., F_D\}$ of features associated with each grid point in ϕ (i.e. each grid point is of dimension D). As described in Chapter III, the elements in Φ can include velocities, temperature, density, mass fractions, progress variable, or any other quantities of interest. In this work, K-means clustering is performed over the set Φ. In non-hierarchical hard-clustering algorithms such as K-means, the three facets that govern the clustering output are 1) how the grid point is represented via $\mathcal{F}$, 2) how the similarity between two grid points is defined (through a distance function), and 3) the number of clusters, K.

The K-means algorithm (Alg. 1 described in Sec. 3.2) upon convergence produces a set of *centroids* $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$ and *labels* for each grid point in Φ contained in the assignment matrix $\mathbf{L}$ (Eq. 3.4). Recall that these centroids are statistical quantities that are similar to the grid points in that they are also made up of D features. A subset of grid points in Φ which have similar feature values is represented by the same centroid. The centroid that represents a grid point is the one that is the closest to the grid point based the L_2-norm in the $\mathbb{R}^D$ composition space. As such, the grid point is labeled to cluster k if it is closest to centroid k in the Euclidean sense. This restricts each grid point in $\mathcal{S}$ to be labeled with only one centroid. The K *clusters* can then be visualized in physical space through the grid point labels, as all members in Φ which share the same centroid label occupy the same cluster. Further, the same set $\mathcal{C}$ produced from one dataset can be used to classify any other dataset so long as each grid point in this new dataset contains the same number of features D as the original data used to generate the centroids. In the results below, the clustering generated by the training set (LMDE described in Sec. 4.2) will also be extended to the testing set (stratified fuel-air mixture) to validate the physical significance of the clusters.

It should be noted that the selection of features $\mathcal{F}$ requires some level of physical intuition by the user. In this application, the features should properly characterize differences in regions of differing chemical mechanisms in the detonation and deflagration regimes. With this in mind, $\mathcal{F}$ should contain at minimum temperature, pressure, and intermediary species as these are the primary drivers of the combustion chemistry in the computation of reaction source terms and in the determination of local flow enthalpies. Thus, $\mathcal{F} = \{\rho, T, P, Y_H, Y_O, Y_{H_2O}, Y_{OH}, Y_{HO_2}, Y_{H_2O_2}\}$ is selected as a starting point, producing $D = 9$. This set should be treated as an upper bound on the total "required" number of elements in $\mathcal{F}$ – it is not guaranteed that the contribution of each feature to the resulting classifications is equal. As such, to reduce the level of supervision required in feature selection, a downselection procedure for $\mathcal{F}$ using an importance metric is provided in Sec. 4.3.4.

As described in Chapter III, the selection of the number of clusters, K, is nontrivial and can vary based on the clustering application. Broadly speaking, the number of clusters should be high enough such that there is sufficient resolution of the macroscopic regions in the domain and low enough such that there is a sufficient number of grid points per cluster. Note that although there are two overarching regions of interest in the domain (detonation and deflagration regimes), this does not necessarily translate to the selection of $K = 2$. Since the level of chemical complexity in these regimes is not the same, a higher level of refinement is expected to establish finer distinctions between detonation and deflagration. Through a user-guided inspection, it was found that cluster numbers beyond 6 for the dataset used here did not result in significant differences in the delineation structure. As such, all of the clustering results below are shown for $K = 6$. Further detail regarding the subtlety of K selection is provided in Chapter III, Sec. 3.2. Methods to guide cluster number selection based on hierarchical cluster refinement strategies are provided in Chapter V via physics-guided K-means extensions.

4.3.2 Clustering Results

The clustering procedure was carried out on the training set for $K = 6$. Before proceeding with the physical interpretation of the clusters themselves, an illustration of the way in which the resulting delineations can be visualized is shown in Fig. 4.4. The grid point assignments of one training snapshot for each of the 6 clusters is shown on the left. Each delineated region represents a single cluster whose grid points are colored in white. Figure 4.4 shows that 1) some clusters represent more grid points than others, and 2) the cluster structures preserve a large amount of coherence in space. These two points will be assessed further below. Additionally, since these grid point labels are non-overlapping (a grid point is only assigned to one cluster), all of the cluster labels can be consolidated into a single image called the *segmented field*, shown on the right of Fig. 4.4. Such segmented flowfields provide a concise juxtaposition

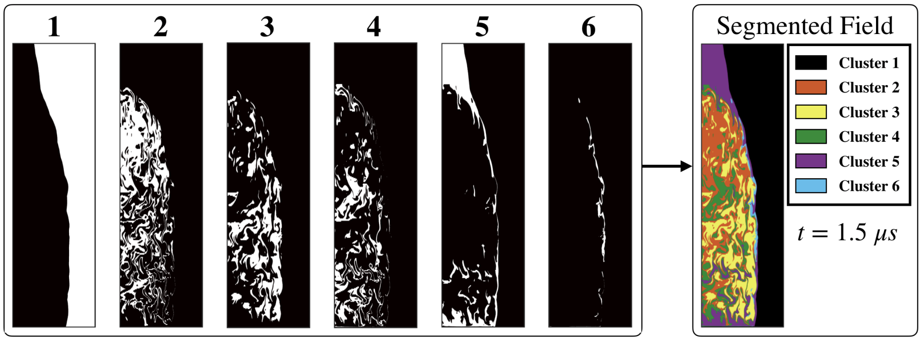


Figure 4.4: (Left) Grid point labels for the 6 clusters for one snapshot, where white regions correspond to cluster assignment. (Right) Combination of the labels into segmented field, where different colors indicate clusters.

of the various cluster patterns and will be used to facilitate the discussion of the clustering output below.

A summary of the labeling results as produced by the clustering is provided in Fig. 4.5 for both training (left) and testing (sets). In each of the training and testing sets, three snapshots are used to concisely illustrate the effect of the time-evolution of the segmented fields. Pressure and temperature fields at the same time instances are also shown. It should be noted that although only three training snapshots are shown in Fig. 4.5, all 13 were used to produce the K-means output.

From the training data results in the left of Fig. 4.5, it can be seen that the selection of $K = 6$ produces distinct clusters corresponding to the different flow states within the training data domain. Cluster 1 corresponds to the ambient mixture region of varying equivalence ratio at 0.5 atm and 297 K. Cluster 5 signifies the shock-separated region where the lack of fuel causes the detonation wave to transition to a leading pressure wave followed by a lagging region of high-temperature products. Furthermore, these regions appear along the entire detonation wavefront, highlighting the finite distance behind the shock front before the gas mixture begins to react. Thus, cluster 5 represents a region where the gas has been compressed but is largely non-reacting. The shock wave acts as a near-sonic throat, where past analysis has shown that the Mach number is approximately 1.3 [273]. Cluster 6, the smallest of all clusters (in terms of proportion of represented grid points) is classified in regions around and including the triple points at the detonation wavefront.

The triple points are concentrations of high-pressure and temperature, resulting in high heat release rates. Along the shock front, sections of the wavefront where cluster 6 is observed signify strong detonation, where the leading shock wave and the reaction zone are closely coupled. Here, the profile of the thermodynamic parameters across the wave closely resembles the ideal ZND profile. It is important to note that cluster 6 exists only in parts of the wavefront. Surrounding cluster 6, regions identified by cluster 5 signify shocked gas, either with entrained non-reacting gas or gas undergoing the induction process. Interestingly, the triple points highlighted by cluster 6 appear in the transition to the shock-separated region, becoming progressively smaller as the pressure concentrations are dissipated within the upper portion of the frame. Thus, the three-dimensional detonation wave structure varies spatially along the front, leading to a corrugated temperature and pressure profile. As the detonation wave moves through the partially-premixed distribution of fuel and air, triple points propagate along the wave front. As they near regions of non-detonable mixture, the pressure concentrations diminish, such as near the shock-separated region and base of the channel. The propagation of these triple points is crucial to maintaining wave strength and the trailing transverse waves form vortical structures that support post-detonation homogenization.

Following the detonation mode, combustion transitions to deflagration represented by clusters 2, 3, and 4. Here, the post-detonation gases burn at high temperatures, rendering temperature a main driver for the distinction between different combustion regimes. Due to post-detonation expansion, temperature and pressure decrease. The separation of the detonation front above the fill height of the injectors, highlighted by cluster 5, creates a shear layer that bisects the wavefront. Post-detonation gases travel upwards behind the wavefront and are turned by the shear layer, increasing the mixing of residual gases. It is interesting to note that this results in more complete combustion of gases. The post-detonation region is identified by cluster 3, where high temperature deflagration and gas expansion supports detonation wave propagation. However, regions of slower heat release within clusters 2 and 4 extend farther behind the wave front. The post-detonation region within the primary fill height of the injected mixture is highlighted by cluster 3, signifying a more ideal detonation process within this height range. Cluster 2 exists within regions of lower temperature deflagration and increased mixing enforced by the shear layer at the fill height. The base of the channel where regions of air in between the injectors are entrained is highlighted by cluster 5 (shocked gas) and cluster 4 (weaker, low temperature deflagration). Relating the temperature and pressure contours to the classified regions, deflagration is captured primarily by clusters 2 and 3 whereas cluster 4 represents the transition to cluster 5.

The application of the learned classifications to the testing data (right of Fig. 4.5) essentially shows where the same features/regimes recovered from the training LMDE dataset (the 6 clusters) are located in the stratified testing case. Interestingly, the representation of the features along the detonation wavefront are quite similar in physical significance to that of the training set, which validates the above analysis. For example, strong detonation where the shock and reaction fronts are closely attached occur only in areas where fuel exists, denoted by cluster 6 along the wave front. Cluster 5 is entrained within the post-shock region due to regions of non-detonable gas processed by the wave front that is characterized by high pressures due to shock compression. In the training dataset, cluster 5 was visible near the fringes of the detonable mixture. Here, the compressed gas exists throughout the post-detonation region in regions roughly corresponding to the size of the stratification. Cluster 3 represents high-temperature deflagration and gas expansion in the post-detonation region, where a slower heat release process consumes the residual reactant mixture.

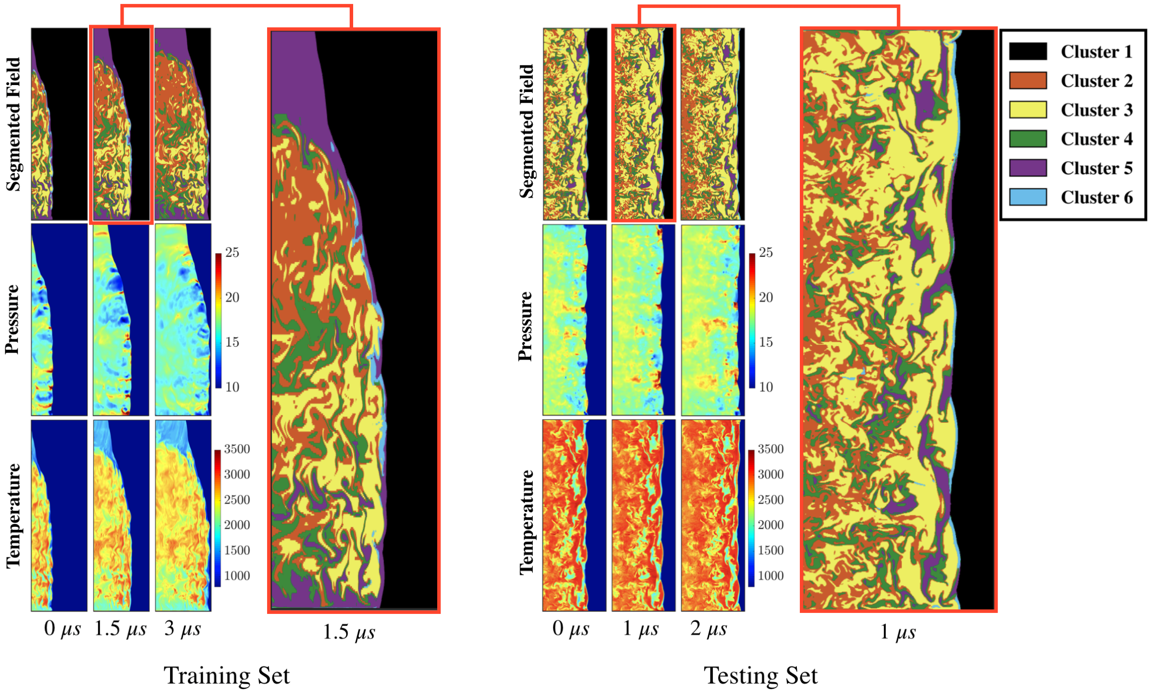


Figure 4.5: Spatial classification of the detonation wave in (left) the training and (right) testing datasets. Shown is the time progression for each configuration given by a smaller subset of the available snapshots. The evolution of the segmented field is compared to the pressure (atm) and temperature (K) distributions. A blow-up of an intermediate snapshot is provided in each case for clearer visualization of the segmented field.

The eddy structures stemming from the collision of triple points and transverse waves ensure that the partially burnt gases are consumed through the deflagration process. Similar to the training dataset, clusters 2 and 4 exist further from the wave front. However, as the wave front is largely perpendicular to the propagation direction, the transition between clusters 2, 3, and 4 is more evident. Cluster 4 exists closer to the wave front and may be characterized by lower temperature deflagration. The appearance of cluster 2 at distances far from the front signify that this region represents a more homogeneous mixture where the reacting gas has reached some equilibrium condition. This further qualifies that the regions represented by cluster 2 in the training dataset may represent a more homogeneous mixture due to the mixing enforced by the shear interactions at the injector fill height.

4.3.3 Analysis of Time Evolution of Segmented Field

The above discussion assigned physical significance to each cluster in the context of the training data, and then extended the findings into the testing set. To assess the effect of time progression on the segmented flowfield, Fig. 4.6 shows the time evolution of the *cluster size* for both training (solid lines) and testing (dashed lines) datasets. The cluster size is defined as the proportion of total number of grid points per snapshot occupied by a particular cluster – as such, for each time instance, the cluster sizes sum to unity. To see which clusters dominate at a particular time instant, the absolute cluster sizes are shown in the left plot of Fig. 4.6. To better represent how cluster sizes evolve, the right plot shows the absolute sizes normalized by the initial sizes at $t = 0$. Since the clusters themselves have physical meaning, the evolution of cluster size is useful metric when combined with the physical space representation in Fig. 4.5 in that it both 1) provides a coarse-grained picture of the time evolution of the detonation structure and 2) allows for the comparison of this structure evolution across different datasets.

From Fig. 4.5, the spatial structure of the delineated clusters is maintained with time progression, with introduction of clusters 2 and 4 on the left-hand side far from the wave front as the detonation wave passes to the right. Regions identified by clusters 2 and 4 are coherent between snapshots as they convect behind the wave front. However, there is transition between clusters 3 and clusters 2/4 as the mixing behavior in the post-detonation region changes with time. The triple point propagation is identified by the movement of cluster 6 with time in a direction along the wave front surface. Thus, the clustering process identifies and tracks the movement of the triple points with reasonable accuracy. Most importantly, the cluster classification is consistent between snapshots, allowing for each cluster to be interpreted with some physical relevance to flow structures with time.

Figure 4.6 displays trends that are both shared different across both datasets.

An expected result is the decrease in size of cluster 1 (the ambient region) in both datasets – as the wave progresses through the domain, the ambient domain proportion is reduced with time. Further, the identification of the triple point regions, cluster 6, is nearly consistent with time. This is an important result as this region occurs primarily near the shock front. As a fixed portion of the detonation front exists within each snapshot, the total number of points corresponding to this cluster remains stable. For both the training and testing dataset, cluster 6 is the lowest populated cluster as the triple points are concentrated regions due to wave interactions. In the training dataset, the size of the post-detonation cluster 3 along with deflagration clusters 2 and 4 increase in time. However, as cluster 3 was observed to be contained to a finite region behind the wave front, the cluster size increases slightly as the detonation wave fully enters the frame and reaches a stable size - this is evident in the normalized cluster size evolution (right plot of Fig. 4.6) of the training dataset. On the other hand, the sizes of cluster 2 and 4 increase a greater rate, with the increase in cluster 2 most dominant for both the training and testing dataset. This is expected, as the post-detonation regions represented by cluster 2 increase in size to match the reduction in cluster 1. Cluster 5 increases rapidly for the training dataset whereas this region is largely constant for the testing set; the shock-separated region increases as the wave progresses to the right in the LMDE, but in the stratified gas case, the amount of entrained high-pressure gas is consistent prior homogenization due to post-detonation mixing.

The rate of change in cluster size in the training dataset is more drastic in comparison to that of the testing set as the the training set features distinct regions, such as the shock-separated region, that are enforced by the fuel-air distribution due to injection. In the testing set, the stratification ahead of the wave is distributed more randomly, and does not ensure one cluster will dominate within different locations perpendicular to the shock front. Ultimately, the interpretation of time-evolution of

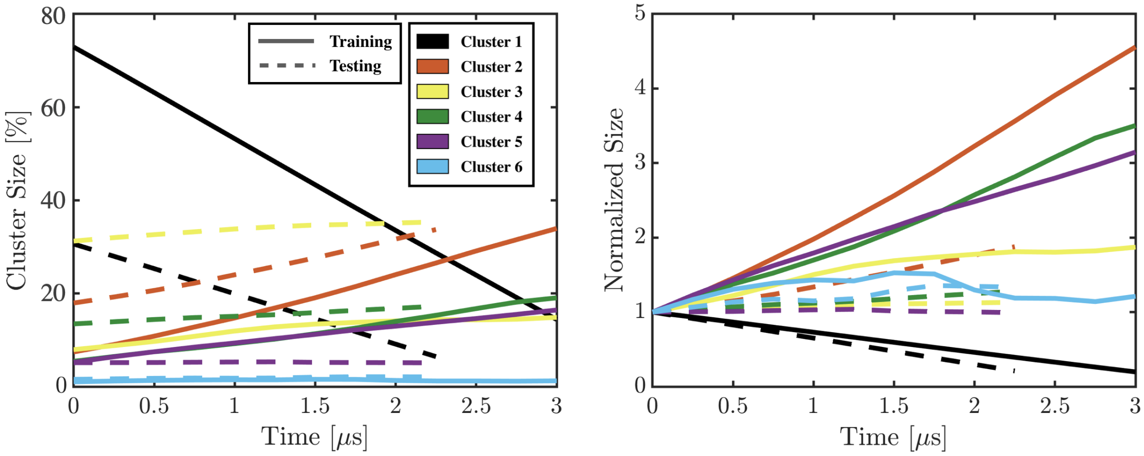


Figure 4.6: Absolute cluster sizes as percentage of total grid points (left) and cluster sizes normalized by the sizes at $t = 0$ (right). Colors represent different cluster numbers. Solid lines correspond to training set, dashed to testing set. Note that lines corresponding to testing set end at a lower maximum time (less testing snapshots).

cluster size allows for a unique and useful way to a) assess a coarse-grained surrogate of the evolution of the full detonation wave structure, and to b) compare the evolution across different detonation configurations.

4.3.4 Feature Importance in Flowfield Classification

Following the above macroscopic analysis of the physical representation and time evolution for each cluster, the discussion below explores the finer details related to conditional distributions of grid points within each cluster. In particular, the information provided by the various features as it relates to the output segmented fields is analyzed, and a pathway for quantifying feature importance and downselection (or pruning) is provided.

Figure 4.7 shows typical pressure versus inverse-density relations for the training set. The points corresponding to each cluster are identified by their respective colors, and the centroids are indicated by the larger white-enclosed markers. An expected trend is observed for cluster 1, which is an outlier in this space since this cluster represents ambient chemical conditions. Some separation is seen along the density axis for

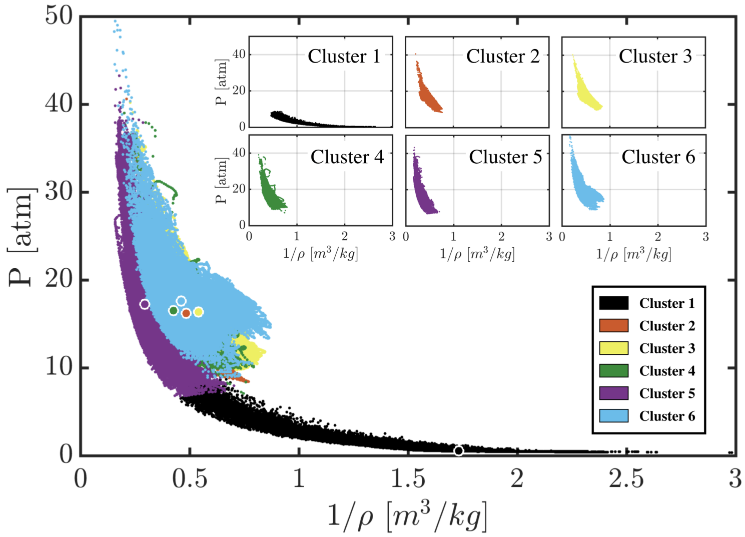


Figure 4.7: Representation of pressure versus inverse-density for all clusters. The larger points with white outline indicate centroids. Insets indicate cluster-specific scatter plots. Colors indicate cluster number.

cluster 5, whose centroid occupies the highest density value. This is evidenced by the its spatial distribution in Fig. 4.5, which revealed that this cluster strongly represents a high-compression minimum-reaction region. Interestingly, the peak pressures near 50 atm are realized by points belonging to the cluster 6 (the region near the wavefront affected by triple-point structures). However, the presence of overlapping densities in the pressure versus inverse-density space (especially for clusters 2, 3, 4, and 6) is telling with regards to detonation wave structure classification: the primary role of the pressure and density features lies in 1) creating the delineation between reacting and ambient portions of the domain, and 2) identifying cluster 5 as a particularly high-density region. Ultimately, the remaining 7 features must contain the additional contribution to the variation in cluster-based probability densities of the grid points for the production of the segmented field structures. In other words, many more axes must be added to Fig. 4.7 to fully explain the regime delineations produced in Fig. 4.5.

The above discussion regarding Fig. 4.7 necessitates a more complete analysis of feature contribution to the overall delineation output. This relies on the interpretation

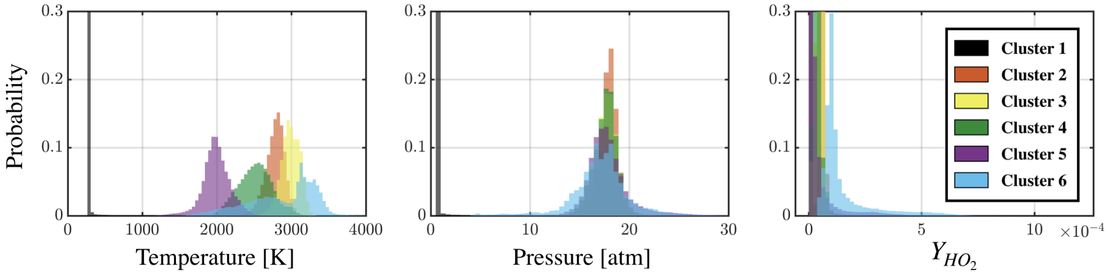


Figure 4.8: Cluster PDFs conditioned on temperature (left), pressure (middle), and Y_{HO_2} (right) for the testing dataset (trends for training set were identical).

and manipulation of probability distribution functions (PDFs) conditioned on both cluster number and feature. To illustrate this, Fig. 4.8 shows cluster-conditional PDFs for three of the nine total features: temperature (left), pressure (middle), and Y_{HO_2} (right). Each color corresponds to the grid point distribution in a single cluster. By analyzing such PDFs for each feature, one can assess how effective each feature is in determining the differences between grid points belonging to different clusters. If cluster PDFs for a feature are similar (in both mean and variance), then that particular feature contributes less to the distance measure used in the clustering. Therefore, the degree of separation, or dissimilarity, between cluster distributions in Fig. 4.8 is related directly to the "importance" of those respective features in the segmented field representation. As expected, in both temperature and pressure distributions, cluster 1 is the outlier. Of the non-ambient clusters, there are much greater differences in distribution in for temperature than for pressure (though the pressure distributions still display differences in variation about the mean). Thus, for these clusters, it can be surmised that temperature is a more contributing feature in determining the delineations shown in Fig. 4.5, in particular for regions within the deflagration regime. On the other hand, for Y_{HO_2}, the PDFs for all the clusters are much more stacked, implying lower overall delineation contribution.

The analysis of the distributions in Fig. 4.8 allows for a qualitative assessment of feature importance in the classifications shown in Fig. 4.5 based on cluster distributions. For a more quantitative analysis, a PDF-based measure known as the Earth Mover's Distance (EMD [288]) can be used to create an importance metric for the features. The EMD is defined for two PDFs (which can be empirical) over a specified domain support, and is given by $\mathcal{D}(p_1, p_2)$, where p_1 and p_2 are PDFs, and $\mathcal{D}$ is the EMD function. The following distance properties hold: $\mathcal{D}(p_1, p_2) \geq 0$, $\mathcal{D}(p_1, p_1) = \mathcal{D}(p_2, p_2) = 0$, and $\mathcal{D}(p_1, p_2) = \mathcal{D}(p_2, p_1)$. A summary of the mathematical formulation and definition is provided in the Appendix. Informally, if p_1 and p_2 are visualized as piles of dirt, $\mathcal{D}(p_1, p_2)$ represents the "cost" of morphing one pile of dirt into the other. The factors which contribute to this "cost" are 1) the distance required to move the dirt, and 2) the amount of dirt present in the movement. The EMD is an appropriate measure here over other PDF-based distances such as Kullback-Leibler (KL) divergence, as it is well-defined for PDFs with nonzero densities in the feature space. The EMD usefully takes into account differences in distribution means as well as variance when determining the final dissimilarity score (the EMD of two identical distributions centered around different values will result in a positive distance, as will the EMD of two distributions with the identical centers but different standard deviations). Thus, using the EMD, a quantifiable metric for feature importance, denoted $\mathcal{I}(F_i)$, where F_i is the i-th feature in $\mathcal{F}$, is defined as

$$\mathcal{I}(F_i) = \sum_{j=1}^{K} \Big(\sum_{k=1}^{K} \mathcal{D}(p_j^{F_i}, p_k^{F_i}) \Big), \quad i = 1, ..., D. \tag{4.1}$$

In Eq. 4.1, $p_j^{F_i}$ represents the PDF of the j-th cluster corresponding to feature F_i. The importance metric is essentially a sum of the EMD combinations of cluster distributions for a particular feature. The inner summation in Eq. 4.1 represents the contribution of j-th cluster to the overall importance. By this metric, if feature 1 has lower importance than feature 2, feature 1 is less significant overall in the clustering output, i.e. it is less crucial in illuminating differences between grid points belonging to different clusters. This metric can be used to effectively downsample the feature selection used to generate the labels in Fig. 4.5.

Figure 4.9 shows importance metrics for each feature in the training and testing datasets. Note that scaled quantities were used in the computation to allow for comparison of importance across different features. The colors in each bar correspond to the cluster contribution to the importance of that particular feature, which itself can be useful in the identification of feature-regime relationships. For example, in the case of Y_{H_2O}, cluster 3 (active predominantly in the deflagration regime) contributes a large amount to the importance. On the other hand, for Y_H and Y_O, cluster 6 (active near the wavefront and triple point regions) dominates the importance value. Thus, an ability to quantitatively assign a most "relevant" cluster to a particular feature in the context of delineation power can be obtained.

Interestingly, Fig. 4.9 shows that the distribution of importance is similar between training and testing sets. In both cases, the metric implies that Y_{HO_2} and $Y_{H_2O_2}$ are noticeably the least significant in the clustering. To illustrate the utility of the EMD-based metric, the clustering is performed again using a reduced feature set *without* Y_{HO_2} and $Y_{H_2O_2}$. These results are shown in Figs. 4.10a and c for a single training and testing snapshot respectively. Clustering using the downsampled set of features results in a segmented field that is nearly identical to that generated from the original, larger feature set. The same characteristics of the detonation wave and the regions of interest are captured by the reduced feature set. Thus, the importance metric provides useful insight into the necessary thermodynamic properties and species that demarcate the different modes of combustion.

The comparison with the original and reduced feature sets is also shown in Figs. 4.10b and d. These are distance plots, which is a proxy measure of uncertainty in the labels used to generate the segmented fields. They depict the distances of each grid point to its assigned centroid; high values in the distance field correlate directly with high

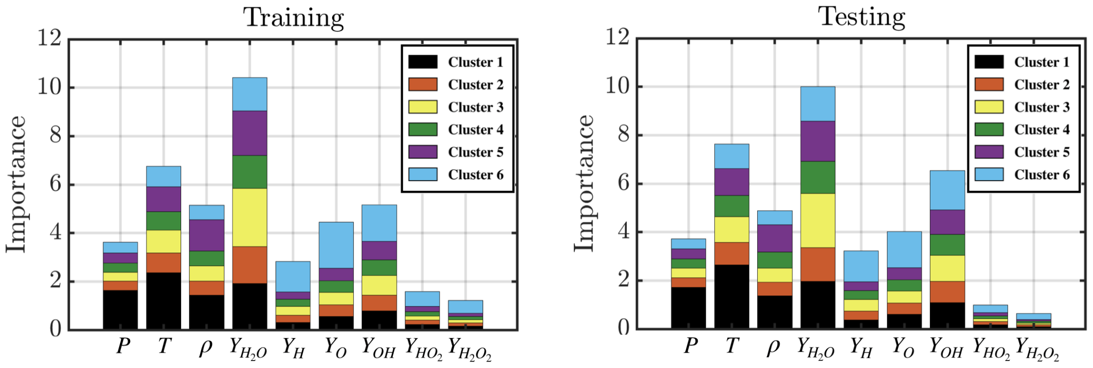


Figure 4.9: Feature importance measures with cluster contributions for the training (left) and testing (right) datasets. Cluster contributions for importance value for each feature by respective colors.

classification uncertainty. In both training and testing sets, points of highest distance occur near the triple points. This means that despite the fact that the presence of the triple points is captured within clusters 5 and 6 to an acceptable level, the uncertainty in classification near the triple points is relatively high with the chosen parameters. This is an indicator to the chemical complexity in this portion of the detonation wave. However, alongside this, an important note is that this distance field is practically unchanged when clustering with the reduced feature set (i.e. the removal of Y_{HO_2} and $Y_{H_2O_2}$ did not cause noticeable increase in label uncertainty), meaning that the clustering output as a whole has been preserved in the process of reducing the feature set. For this reason, it is important that the initial classification is performed with a full (or very large) feature set available from the data in general applications. Using the original clustering followed by application of the importance metric, a lower feature set can then be obtained to identify both redundant and important regime-dependent features and also to reduce computational cost for in labeling for potential online applications.

Ultimately, the discussion in Sec. 4.3.2-4.3.4 provided a physical interpretation of the delineations produced by the clustering, as well as a pathway for feature downselection based on the EMD importance score and preservation of distance fields. With

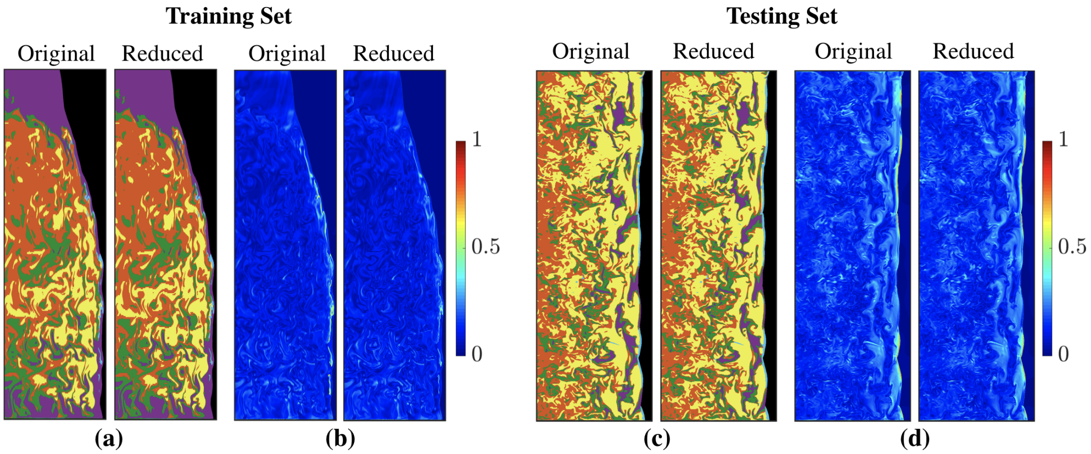


Figure 4.10: Comparison of the clustering output (only one snapshot shown) between original full feature set and the reduced set. (a) Segmented fields for training dataset. (b) Distance fields for training dataset. (c) Segmented fields for testing dataset. (d) Distance fields for testing dataset.

this context, the localized source term modeling strategy conditioned on the composition space partition produced by K-means can now be performed.

4.4 Source Term Regression

In this section, the delineated regions for $K = 6$ obtained in Sec. 4.3 are used to guide the modeling for thermochemical source terms of interest. The ANN-based source term modeling approach is of particular interest here since the relations between the thermochemical state and the source terms are highly nonlinear and render computationally intensive chemistry routines intractible in reacting detonation solvers. The goal is to show that proper utilization of the segmented fields produced in Sec. 4.3 can lead to more robust modeling procedures that better enable long-time simulations. Specific details into ANN methodology are omitted here, but can be found in Refs. [108, 214].

4.4.1 ANN Architecture

Two types of ANNs are considered: global and local. The global ANN is trained agnostic to the cluster labels and uses the full set grid points as the training data. In contrast, the local ANN refers to the model trained only for data belonging to cluster k. This means that there are a total of 1 global ANN and 6 local ANNs (1 for each cluster). This is similar to the method used in Ref. [22] to display the advantage of domain-localized modeling, but the difference here is that the domain localization is learned via the clustering step. Throughout the results below, local versus global ANN prediction accuracy for the source terms (conditioned on cluster number) will be discussed.

All ANNs contain two hidden layers, each with 50 neurons. The hyperbolic tangent function is used for hidden layer activations. As mentioned in the end of Sec. 4.2, in the ANN implementation, the training set is restricted to a single snapshot from the LMDE simulation. The testing set contains a snapshot from the stratified fuel-air mixture case. Additionally, to assess model performance in both similar and different configurations as the training data, the testing routine also consists of unseen snapshot from the LMDE configuration at a later timestep than the training set. These training and testing snapshots are shown in Fig. 4.11a.

The full ANN architecture is visualized in Fig. 4.11b. The ANN inputs are the same as the *reduced* feature set defined in Sec. 4.3.4; that is, all input grid points to the ANNs are described by $\{\rho, T, P, Y_H, Y_O, Y_{H_2O}, Y_{OH}\}$. The output source terms are given by $\{\dot{T}, \dot{Y}_H, \dot{Y}_O, \dot{Y}_{H_2O}, \dot{Y}_{OH}\}$. All inputs and outputs were standardized before training. Gradients of an MSE loss function are found using backpropagation, and the Adam algorithm was used for parameter optimization [125, 151]. Training samples were randomly shuffled with 10% of the dataset set aside for validation to monitor overfitting. The neural networks were implemented with the PyTorch library [242].

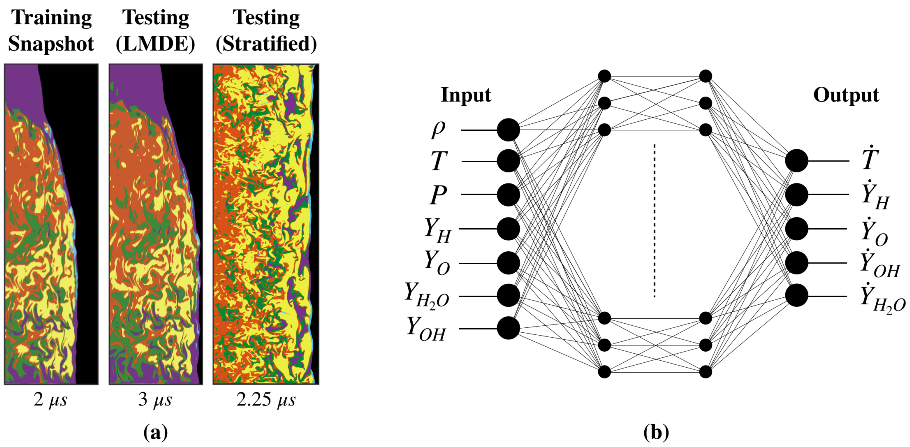


Figure 4.11: (a) Representation of the training and two testing snapshots for the demonstration of ANN source term regression (corresponding time instances are given at bottom). (b) Illustration of ANN architecture, where the input corresponds to a grid point represented by the reduced feature set as described in Sec. 4.3.4, and the output is a set of source terms for the same grid point.

4.4.2 ANN Results

The MSE values obtained after completion of the training procedure for both local and global ANN models are shown in Fig. 4.12. In particular, the MSE values for the LMDE training set (top row), LMDE testing set (middle row), and stratified mixture testing set (bottom row) are compared for each of the five source terms and are conditioned on the cluster number (the columns in Fig. 4.12). Note that since the primary purpose of Fig. 4.12 is to show the improvements provided by the local ANN models with reference to the global model, the MSE values are derived from the standardized representation of the source terms to enable cross-feature comparisons. Further, the output for cluster 1 is excluded for the sake of brevity as it represents an ambient uninteresting region with regards to source term output.

For the training set, the MSE values indicate general improvement provided by the domain-localized modeling. For example, in clusters 5 and 6 (which represent the complex near-wavefront regions as shown in Fig. 4.5), the local ANNs display significantly lower errors when compared to global counterparts. This is less pronounced for the deflagration clusters 2, 3, and 4, although improvements provided by the localized modeling is still seen in these clusters for all tested source terms. Further, the MSE of the source term predictions for $\dot{Y}_{OH}$ in the training set are significantly higher than the rest for most clusters in both local and global ANN settings (especially in cluster 4).

The middle row of Fig. 4.12 shows the respective MSE comparisons for a future snapshot (the last of the available 13 as described in Sec. 4.2) in the *same* configuration as the training set (the LMDE testing set). It is convincing that the overall trends are preserved for the unseen data. Interestingly, in clusters 2, 3, and 6, the local ANN MSE values are lower than those observed in the training set for all source term features. This drop in error for the LMDE testing set is not observed in cluster 5, which indicates that the extrapolative power is not distributed uniformly throughout cluster index. However, the fact that significant improvement provided by the localized models is still seen across the board is a form of confirmation of the similar trends observed in the training set. The bottom row of Fig. 4.12 shows similarly structured MSE plots, but for a testing snapshot (the last of 10 snapshots as described in Sec. 4.2) from the stratified mixture configuration. In this testing set, MSE values are significantly higher as expected – the configuration and spatial distribution of the detonation structure are quite different from the training set, and as such, extension of the ANN models to this setting becomes more difficult. Despite this, the MSE trends again show comparatively much more accurate predictions generated by the localized ANN models, especially in clusters 5 and 6. It should be noted that even though such improvements are provided by the local models, instances of significantly high error are seen again for $\dot{Y}_{OH}$ in clusters 2 and 4 in the stratification testing set, even for the local ANNs.

The MSE plots in Fig. 4.12 were obtained by averaging error quantities over all grid points – to visualize performance in a more direct sense, scatter plots of

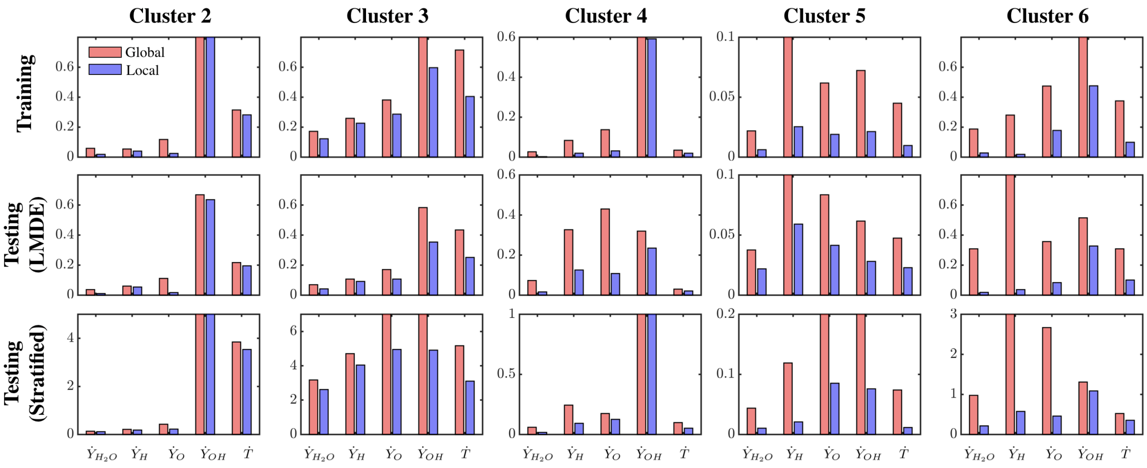


Figure 4.12: Comparison of MSE values for each cluster for global (red bars) and local (blue bars) ANN source term predictions. First row corresponds to the training LMDE dataset, second row to the testing set with the same LMDE configuration but at a future timestep, and last row to the testing set in a stratified fuel-air mixture configuration. Within each plot, MSE is compared over all five features as listed at bottom. MSE is computed over standardized outputs to better facilitate comparisons across multiple features.

predicted source term values versus the ground-truth are shown in Fig. 4.13 for $\dot{T}$ and $\dot{Y}_{H_2O}$ outputs (other source term outputs excluded as all relevant trends are readily identified by these two outputs). Predictions for both testing sets (LMDE at future timestep and stratified mixture cases) are shown. Here, source term values in each of these plots have been unscaled, and the dashed unit-slope lines represents an ideal prediction.

The topmost set of plots in Fig. 4.13 show the results for the LMDE testing set (same configuration as training set, but at a future time instance). The reduction in variance of predicted quantities around the exact solution can be seen when considering the local ANN models – the largest improvement is observed for cluster 6, which represents the region near the detonation wavefront and triple points. However, the improvements provided by the localized modeling are more readily seen for $\dot{Y}_{H_2O}$ predictions than for the $\dot{T}$ counterparts. For example, in cluster 6, the local ANN is slightly better at capturing the negative temperature source term values than the global model, but is still overall inaccurate in this region. Further, consider the $\dot{T}$ predictions for cluster 2 versus cluster 3. The local model fails to address negative values in cluster 2; in cluster 3, though still not perfectly accurate due to the innate complexity of temperature source term distributions in detonating flows, the localized modeling better resolves the negative temperature source term than the global counterpart. In clusters 4 and 5, predictions for both local and global models are accurate in the LMDE testing set and follow the exact line quite well, though the local models observe less variation around the exact solution.

The bottom set of plots in Fig. 4.13 show the results for the stratified fuel-air testing case (different configuration from the training snapshot). As implied by Fig. 4.12, both global and local model performances are altogether poorer than those seen in the LMDE testing set, though this performance loss is much less severe in clusters 4 and 5. Despite the performance drop, improvement is still seen in the local model predictions, particularly for the $\dot{Y}_{H_2O}$ output. For example, in cluster 6, the $\dot{Y}_{H_2O}$ local ANN prediction curtails much of the sporadic variation from the global model. The same is true to a lesser extent in clusters 2, 4, and 5 – cluster 3 predictions, on the other hand, are altogether unsatisfactory, as both local and global predictions show very little correlation with the ground truth. In the predictions for the stratified testing case for $\dot{T}$, noticeable local model improvement is only seen in clusters 4, 5, and 6. The local ANN predictions for $\dot{T}$ in cluster 2 are almost identical to the global ANN predictions, and the same issue with negative temperature source term as discussed for the LMDE testing snapshot is again apparent for the stratified mixture testing snapshot.

From the results in Fig. 4.13, it can essentially be concluded that a) noticeable localized model improvement is indeed observed in most cases, with highest improvements seen for cluster 6, and b) the extrapolation error is much more pronounced in the unseen stratified mixture configuration. A useful summary of these trends

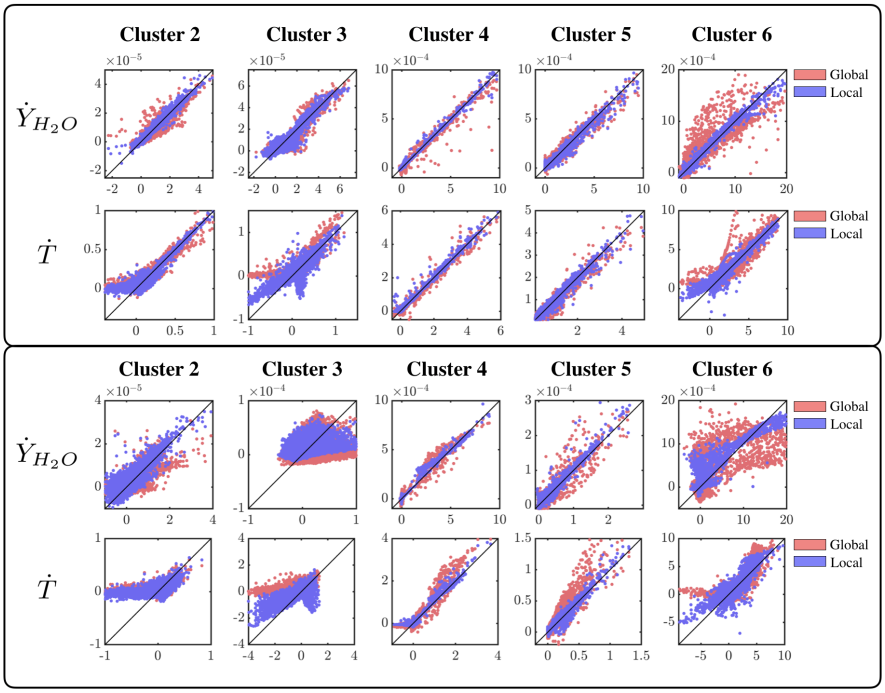


Figure 4.13: Scatter plots showing global (red points) and local (blue points) predictions on the y-axis versus ground-truth on the x-axis for clusters 2 to 6. Diagonal solid black lines correspond to exact solutions. Top block corresponds to LMDE testing snapshot (same configuration as training snapshot) and the bottom block to testing snapshot for the stratified mixture configuration. Within each block, the upper row of plots correspond to $\dot{Y}_{H_2O}$ predictions and lower row to $\dot{T}$ predictions. In these plots, source terms have been scaled by the simulation timestep (same value for all grid points) such that the $\dot{Y}_{H_2O}$ is unitless and $\dot{T}$ is in units of Kelvin.

lies in the visualization of the source term fields themselves, as shown in Fig. 4.14. Fig. 4.14 (left) illustrates the increase in accuracy provided by the local ANN models in the LMDE testing snapshot, though the accuracy gain is more apparent for $\dot{Y}_{H_2O}$ than for $\dot{T}$. In particular, the region above the wavefront (represented by cluster 5 in Fig. 4.5) and the fluctuations of source term behind the wavefront present in the local ANN predictions better resemble the ground truth. On the other hand, the predictions generated for the stratified mixture testing snapshot are much less accurate for both global and local models. Though some regions are better captured by the local models, such as the absence of $\dot{Y}_{H_2O}$ fluctuation in the far-left domain and the general source term structure in the region immediately behind the wavefront for $\dot{T}$, it is apparent that extension of the neural network-based source term models into different configurations is less plausible.

Ultimately, the application of cluster-based localization of source term modeling is quite promising when considering configurations similar to the training set; such restrictions are expected in light of the highly nonlinear nature of chemical source terms in detonating flows. More importantly, this demonstration of localized ANN modeling highlights the potential for in-situ neural network-based source term modeling (where the configuration does not change), a setting in which significant computational savings can be achieved and, as a result, long-time simulations of complex geometries exhibiting detonating reacting flow behavior (such as RDEs) be realized. It is possible that increasing the ANN parameter space, including time-history, including soft physical constraints during the parameter estimation phase, and/or tuning the input/output space for the networks may lead to better results with regards to extension into different configurations (e.g. the stratified mixture case) – all of these comments warrant a more detailed analysis into the localized modeling procedure and will be explored in future work.

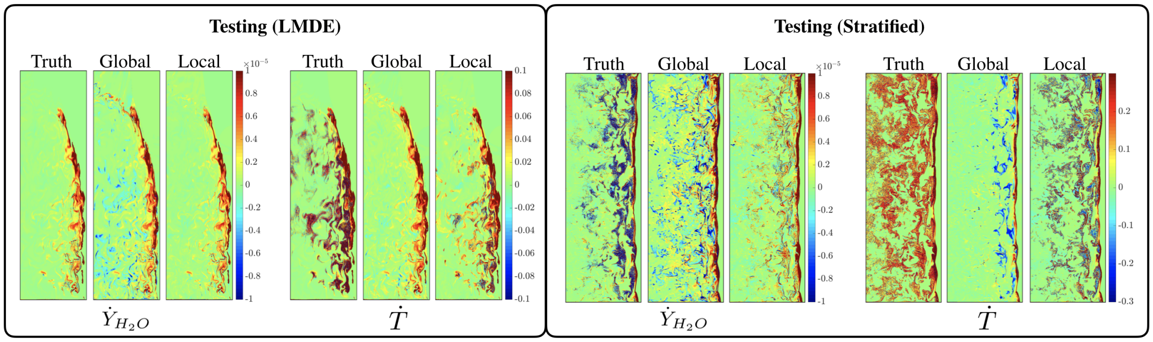


Figure 4.14: Exact and predicted source term fields for the LMDE testing snapshot (left block) and stratified mixture snapshot (right block). Within each block, the left and right group of fields show $\dot{Y}_{H_2O}$ and $\dot{T}$, respectively. In these plots, source terms have been scaled by the simulation timestep (same value for all grid points) such that $\dot{Y}_{H_2O}$ is unitless and $\dot{T}$ is in Kelvin.

4.5 Conclusion

Using direct numerical simulation datasets of canonical detonation configurations, a data-driven modeling procedure was developed with the goal of providing a pathway to better realize long-time simulations for complex combustor geometries such as RDEs. The modeling approach consisted of two linked phases, where the first concerned the extraction of different combustion regimes within the wave structure, and the second concerned the localized development of source term models guided by the extracted regions obtained from the first phase.

Specifically, in the first phase of the procedure (Sec. 4.3), a classification of the flowfield was obtained from a clustering on the progression of a detonation wave through a linear injector array. These clusters represented the ambient fuel-air mixture, a shock-separated region in the absence of fuel, a strong detonation region, and post-detonation deflagration regions within the reaction zone. This assignment of physical representations of each cluster allowed for the development of a useful coarse-grained perspective on detonation chemistry in physical space. Further, the extension of the clustering to the testing dataset (stratified fuel-air mixture) provided a manner in which detonation environments can be compared across different simulation configurations.

A comparison of the source term estimations obtained from the local ANNs (i.e. ANNs trained for each cluster in isolation) with the global ANN counterparts (i.e. a single ANN trained for the whole domain) showed general improvement provided by the domain-localized modeling in the training datasets. When predicting the source terms for unseen detonation waves in the same configuration at a future timestep (the LMDE testing snapshot), the improvements provided by the cluster-localized modeling became especially apparent and promising. When ambitiously extending the trained networks to the unseen data at a different configuration, it was found that although in many regions the localized model alleviated some of the large source term error variation seen in the global model, the source term predictions overall were much poorer. Despite this, the successful demonstration of this cluster-based localization of source term estimation on the unseen LMDE data is promising with regards to the role of domain-localized neural networks in the enabling of long-time simulations for complex combustion chemistry for situations in which the operating configuration does not change.

This chapter illuminates the promising role of data-driven classification and regression techniques both for concisely extracting segmented fields that delineate different combustion regimes and for guiding localized combustion modeling procedures in complex detonating environments. However, there are many promising pathways for future work. For example, for the ANN implementation, including physical constraints and tuning the input/output space for the local models may lead to better results when extending to different configurations. Further, a detailed investigation into computational cost savings is necessary. One of the major motivations for converting the source term computation into an ANN evaluation is the compatibility with GPU architectures (see Sec. 2.3). As such, assessment of the computational savings stemming from both algorithm and architecture change is currently being pursued.

With the core concept of classification-based regression presented in this chapter, the goal of Chapter V is to provide pathways for improved classification by embedding the underlying physical knowledge of dynamics in the composition space (i.e. the functional form of the chemical source term evaluation) into the clustering procedure, such that (a) human-assisted validation steps carried out in Sec. 4.3 required to ensure segmented fields are physically consistent are eliminated, and (b) simpler modeling pathways that allow departure from potentially prohibitive neural network training stages are provided.

CHAPTER V

Physics-Guided Clustering Strategies for Improved Flowfield Classification

5.1 Introduction

As discussed in Chapter I, high-fidelity simulations of next-generation propulsion devices like RDEs and scramjets require resolved numerical treatment of the compressible Navier-Stokes equations with detailed chemical kinetic descriptions to properly account for the coupling of chemical reactions with turbulence and shock waves [258, 276]. Treatment of detailed kinetics in these solvers constitutes a prohibitive computational bottleneck due to the wide range of timescales present in the building-block elementary chemical reactions [30]. The resulting stiffness and nonlinearity stemming from the underlying dynamical system driven by the chemical source term, which comes from a linear combination of these elementary reaction rates, is notoriously difficult to deal with in all reacting flow solvers for three reasons: (1) accounting for flow-chemistry interactions requires solving a stiff ordinary differential equation in the thermochemical phase space for all cells in the computational domain at every simulation time step [258]; (2) the arithmetic intensity of the right-hand-side evaluation for this ODE is very high (i.e. the evaluation of the chemical source term is expensive due to the complexities of the Arrhenius formulation) [16]; and (3) the costs due to time integration in (1) and source term evaluation in (2) scale super-linearly with increases in chemical mechanism complexity [183]. Sections 2.4.2 and 2.5.6 in Chapter II describe existing modeling approaches used within the numerical combustion community to treat these issues, with the ultimate goal of accelerating time-to-solution for the required high-fidelity full-geometry reacting flow simulations.

The classification-based regression strategy presented in Chapter IV tackles the targeted issue of chemical source term evaluation in detonation-containing flows using a data-based localized modeling strategy. The method relies on two steps: (1) identifying regions within the detonation wave structure using a clustering strategy in thermochemical composition space, and (2) deploying ANN based models for source term evaluations within each cluster. The analysis in Chapter IV showed that concept of localized modeling driven by unsupervised K-means clustering improves source term prediction accuracy when compared to a so-called "global" approach that utilizes the same regression strategy (i.e. trained ANNs) without knowledge of the cluster partitions. The method is intended to extend the class of existing data-based chemistry modeling approaches (see Sec. 2.5.6) by building on previous work related to neural network based chemistry tabulation [32, 60] and clustering/partitioning strategies for reacting flow [33, 96].

The motivation in Chapter IV was to utilize the local ANNs as a proxy for GPU-optimal source term evaluation in the hope that they provide sufficient acceleration over conventional analytic routines upon deployment into a flow solver. The problems with ANN-based approaches, however, are the parametrization and training stages. The parametrization stage consists of designing the neural network architecture – although all ANNs are GPU-optimal, they are not necessarily faster than conventional alternatives. The cost of an ANN forward pass scales with (1) the number of hidden layers, (2) the number of neurons per layer, (3) the type of nonlinear activation function, and (4) the input-output scaling procedure. As such, additional steps must be taken to ensure the ANN architecture actually minimizes time-to-solution by comparing hardware-oriented quantities of interest (e.g. floating point operations per second and global memory access times) to an an analytic GPU-optimal baseline, which in-turn requires the development of GPU-based kinetics libraries [16]. Additionally, as described in Chapter IV, the training stage has the tendency overfit to the configuration used to produce the model, limiting extrapolation capability and overall reliability for solver deployment purposes.

The classification strategy in Chapter IV utilizes the standard K-means algorithm in composition space to produce the required flowfield delineations (segmented fields) for localized model deployment. Although the segmented fields can be determined by expert-guided assessment to be correlated to key features such as combustion regimes within the detonation wave structure, there is an inherent issue in the classification approach of Chapter IV from the modeling perspective: in reality, if the source term function is highly nonlinear, there is no reason to believe that clusters produced from standard K-means (which minimize the within-cluster variation in composition space) also minimize within-cluster variation in chemical source terms. This means that conditioning source term estimations on partitions derived from standard Euclidean distances (i.e. standard K-means in composition space) is overall unreliable for complex reacting flows.

The goal of this chapter is to extend the classification-based regression approach described in Chapter IV by embedding physical knowledge in the K-means clustering procedure, which not only addresses the above issue of cluster consistency, but also eliminates the need for ANN-based source term regression. This is accomplished by augmenting the standard Euclidean distance that drives the flowfield partitioning process with the functional form for the chemical source term, creating the class of *physics-guided K-means clustering* methods. Two physics-guided K-means clustering approaches based on the above ideas will be explored and compared:

1. **Jacobian-Scaled K-means (JS-K-means):** As the name implies, the Jacobian-scaled K-means approach directly modifies the Euclidean distance function used in K-means clustering by scaling the distance vector between two points in a given cluster with the Jacobian of the chemical source term evaluated at the centroid. This modification of the distance function induces a new objective function that is different from that of standard K-means; minimization of the new JS-K-means objective pushes clusters towards regions in composition space that reduce within-cluster variation in chemical source term. This method is therefore interpreted as a mechanism for *redistributing* (or biasing) a set of clusters produced by the standard K-means algorithm towards regions of increased dynamical similarity.

2. **Hierarchical K-means (H-K-means):** instead of modifying the standard K-means algorithm directly as in the JS-K-means approach, the hierarchical K-means strategy builds a cluster hierarchy by refining the clusters of an existing K-means partition into sub-clusters. Clusters are flagged for refinement using an error estimation procedure based on similar Jacobian-based scaling of distance vectors. Instead of prescribing the number of clusters K, the refinement procedure terminates until a target error tolerance is met – as such, the requirement of number of cluster specification is eliminated in this approach. As described further in the chapter, the scope of the H-K-means approach is different than the JS-K-means approach described above: instead of *redistributing* existing cluster locations towards regions of physical similarity, the hierarchical approach *adds* clusters in regions of physical similarity by means of an error estimation based refinement procedure.

The above physics-guided clustering algorithms present two separate pathways that arrive at the similar end-result, which is ensuring that the resulting clusters/partitions are intrinsically compatible with the modeling goal (source term estimation). This allows for simpler methods for source term prediction. In other words, because the physics-guided clustering approaches produce partitions that adapt to regions of dynamical similarity, a subsidiary objective of this chapter is to show how source term regression via complex ANN-based frameworks can be traded for simpler, physics-based cluster-dependent matrix-vector multiplications derived from Taylor expansions, where the matrix is the cluster-conditioned chemical Jacobian matrix and the vector is the sample-to-centroid distance vector.

The above methods are demonstrated on canonical detonation datasets sourced from high-fidelity compressible reacting flow simulations. Note that in contrast to Chapter IV, less emphasis in this chapter is placed on the physical interpretation of the clusters in terms of regime identification within the detonation wave structure. Instead, emphasis is placed on (a) the details of the algorithms themselves, as they are novel extensions to the existing class of K-means based approaches, (b) the way in which the partitions produced by the physics-guided algorithms deviate from those produced by standard K-means, and (c) source term prediction quality within the complex regions of the detonation wave (e.g. near the oscillating triple points).

Before proceeding, it should be noted that the general idea of embedding physical knowledge into data-based modeling frameworks for reacting flow is not new. Physics-guided modeling strategies that attempt to tackle chemistry tabulation from a reduction point of view include methods such as ISAT [263] and the PRISM [26] approach. In these methods, the computationally expensive numerical integration of chemical source terms (Eq. 2.43) is replaced by a look-up table. In particular, ISAT builds a trust region in thermochemical composition space using a set of ellipsoids determined by the Jacobian of the source terms, which is similar to the physics-guided clustering approaches presented here. However, in methods like ISAT and PRISM, the cost of building and accessing such tables can become expensive for large mechanisms, especially on modern HPCs that use extensive concurrency in computations to reach high throughput efficiency – the method presented here, on the other hand, attempts to address some of these limitations by leveraging highly scalable K-means algorithms. As many of the core concepts in the physics-guided clustering approach comes from ISAT, comments describing key differences between the approaches are made in more detail in the sections below when describing the respective algorithms.

The following summarizes the remaining sections in this chapter. Section 5.2 describes detonation configuration used to generate the clustering data as well as the flow solver. The Jacobian-scaled K-means approach is then described in Sec. 5.3, followed by the hierarchical K-means strategy in Sec. 5.4. In both Sec. 5.3 and Sec. 5.4, the respective methodologies and algorithms are detailed, followed by demonstrations on simpler toy problems before moving to the more complex detonation dataset. Concluding remarks are provided in Sec. 5.5.

5.2 Description of Data

This section describes the simulation procedure used to generate the clustering dataset. More specifically, details on the simulation configuration, flow solver numerics, and pre-processing steps are provided.

The dataset used in this analysis comes from high-fidelity hydrogen-air detonation simulations in a 2-dimensional channel configuration, as shown in Fig. 5.1. The flow solver is implemented with the finite-volume method in the AMReX library [367], and the governing equations are the compressible reacting Navier-Stokes equations (see Sec. 2.2 for an overview of the PDEs). Although the solver is implemented in the AMReX framework, the simulations conducted for this study do not utilize AMR – all data was collected from single-level runs. Solver numerics are globally 2nd-order and are consistent with those used in UMReactingFlow [30]. Cell-to-face interpolation required to compute numerical fluxes for the advection term is accomplished with a slope-limited TVD method based on the monotonic upstream-centered scheme

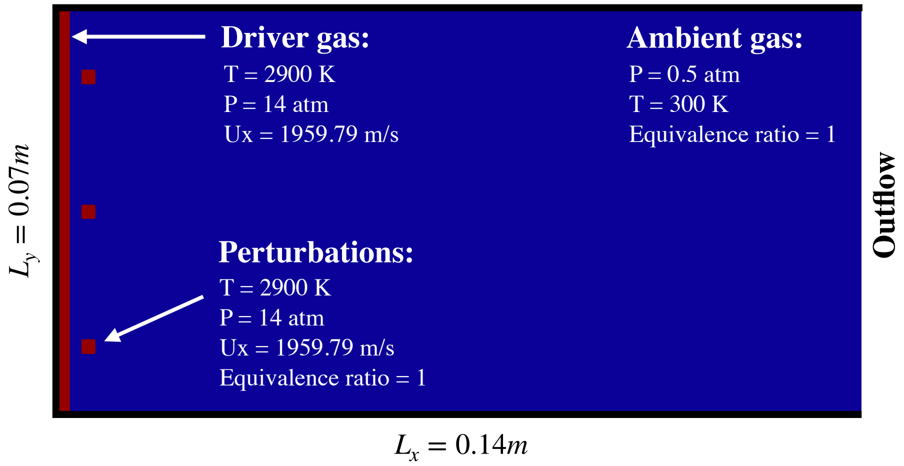


Figure 5.1: Computational domain for channel detonation simulation.

for conservation laws (MUSCL) approach [342]. Modifications to the MUSCL-based interpolation routines, detailed in Refs. [30, 126], ensure that species boundedness is preserved on the cell faces. Advective fluxes are then evaluated with the Harten-Lax-van Leer-Contact (HLLC) approximate Riemann solver [23]. Diffusive fluxes are obtained using standard 2nd-order central schemes. Subsequent surface integration required to compute advective and diffusive flux divergences for each cell utilizes a 2nd-order numerical quadrature. Timestepping is accomplished with a 2nd-order operator splitting method that decouples reactions from the advection and diffusion operators (see Fig. 2.10). Chemical time integration is accomplished with an explicit stiff time integration strategy based on adaptive time stepping within a 2nd order Runge-Kutta scheme. Time integration for advection and diffusion phases is treated with an explicit TVD Runge-Kutta scheme that is also 2nd-order. Chemical kinetics treatment for hydrogen-air chemistry comes from the mechanism of Mueller et al. which consists of 9 species $(H_2, O_2, H_2O, H, O, OH, HO_2, H_2O_2, N_2)$ and 21 reactions [212]. Equipped with this mechanism, the open-source library Cantera is used to evaluate of chemical source terms, transport coefficients, and other relevant thermodynamic quantities [109].

As shown in Fig. 5.1, the channel is 0.07 m in the y-direction and 0.14 m in the x-direction. The boundary conditions consist of slip walls everywhere except for the rightmost boundary, which is an outflow. The unsteady detonation is initiated by prescribing a high-energy driver gas in a 2 mm region on the leftmost side of the domain. The temperature, pressure, and x-component of the velocity for the driver gas is set to 2900 K, 14 atm, and the CJ speed, respectively (the y-component of the velocity is zero everywhere initially). The species mass fractions for the driver gas are obtained from the CJ condition evaluated from the idealized ZND detonation solution (the Caltech Shock and Detonation toolbox was used to this end [312]). Similar to the approach used in Ref. [273], roughly 2 mm ahead of the driver gas, an array of cube-like perturbations of high-energy stoichiometric hydrogen-air mixture is placed slightly 2 mm ahead of the driver gas region. These perturbations are required to initiate transverse wave reflections that produce the characteristic triple point structures observed in unsteady detonations in 2 and 3 dimensions. The composition for the ambient gas is stoichiometric H2/Air at 0.5 atm and 300 K.

The computational domain is discretized with 4096 cells in the x-direction and 2048 cells in the y-direction, producing a total cell count of roughly 8 million and a grid resolution of $\Delta x \approx 34$ micron. At an ambient pressure of 0.5 atm, this grid resolution results in roughly 10 cells within the induction zone and therefore exceeds required spatial resolution limit for resolved unsteady H2/Air detonations [272]. Starting from the conditions specified in Fig. 5.1, the simulation was run for a total time of 10^{-4} seconds at a CFL number of 0.6, which was enough time to ensure progression of the developed CJ detonation through the entirety of the channel.

A subset of the full channel domain at $t = 50$ μs that contains the entirety of the detonation wave structure is extracted in a cropping procedure similar to the one used in Chapter IV. The cropped subdomain (bounded by $x = [0.085, 0.95]$ and $y = [0, 0.07]$) produces a set of $N = 598,015$ thermochemical state vector samples that describe all relevant details of the detonation wave structure (e.g. triple point

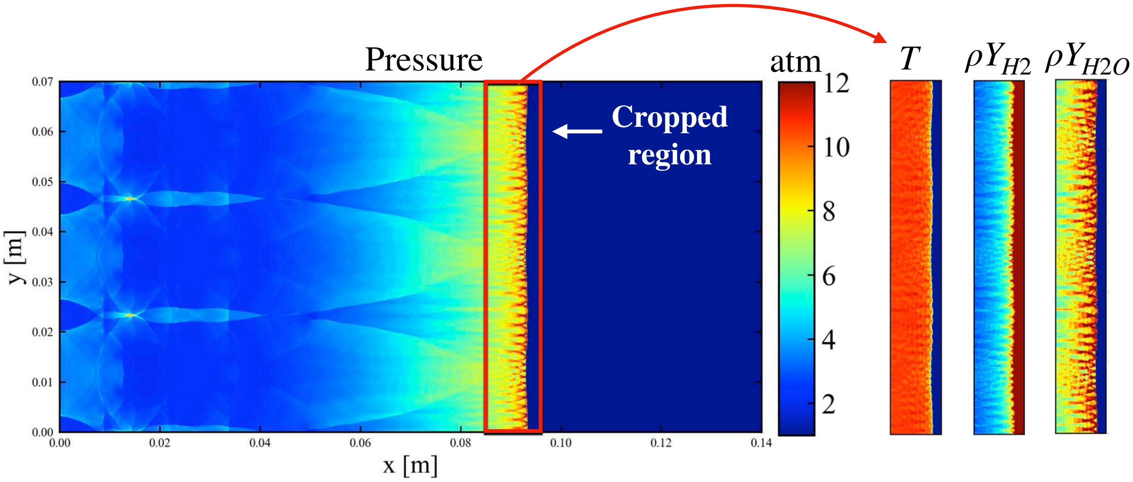


Figure 5.2: Cropping procedure for an instantaneous detonation flowfield at $t = 50$ μs used to generate the clustering dataset. Colorbar ranges for T, ρY_{H2}, and ρY_{H2O} are $[300, 3200]$ K, $[0.001, 0.006]$ kg/m^{3}s, and $[0.1, 0.2]$ kg/m^{3}s respectively.

oscillations, transverse waves, deflagration regions, ambient regions, etc.) and ignores the regions beyond the sonic choke point that are irrelevant to the wave dynamics.

The individual samples are denoted $\phi_i \in \mathbb{R}^D$, where $i = 1, \dots, N$. The ϕ_i reside in the 10-dimensional composition space defined by temperature and species mass concentrations (i.e. $D = 10$). More specifically, $\phi_i = [T_i, \rho Y_{1,i}, \dots, \rho Y_{N_S,i}]^{\mathrm{T}}$, where T_i is the temperature for sample i, ρY_k is the k-th species concentration, and T denotes the transpose operation (not to be confused with temperature). This definition for the composition vector produces a concise representation of the thermochemical state, as the local fluid density can be obtained from a summation over the species mass concentrations and the pressure from the ideal gas law. Using the composition space samples, two datasets are created: the first is the composition data matrix $\Phi = [\phi_1, \phi_2, \dots, \phi_N] \in \mathbb{R}^{D \times N}$ and the second is the ground-truth chemical source term data matrix $\Omega = [S(\phi_1), S(\phi_2), \dots, S(\phi_N)] \in \mathbb{R}^{D \times N}$, where $S_i(\phi) = [\frac{dT_i}{dt}, \frac{d\rho Y_{1,i}}{dt}, \dots, \frac{d\rho Y_{N_S,i}}{dt}]^{\mathrm{T}}$ is referred to as the chemical source term vector for sample i. The formulation for species production rates are provided in Sec. 2.2. The temperature source term is evaluated under the assumption of a constant-volume reactor as

$$\frac{dT_i}{dt} = -\frac{1}{c_v}\sum_{k=1}^{N_S}\frac{\epsilon_k}{W_k}\frac{d\rho Y_k}{dt}, \tag{5.1}$$

where c_v denotes the mass-based specific heat at constant volume for the mixture, ϵ_k is the molar internal energy for species k (derived from NASA polynomials), W_k is the species molecular weight, and $\frac{d\rho Y_k}{dt}$ is the production rate (source term) for species concentration k. Ultimately, the composition data in Φ is used in the physics-guided clustering procedures in Sec. 5.3 and 5.4 and the data in Ω is used to evaluate the predictive capability and errors for the respective source term predictions.

5.3 Jacobian-Scaled K-means Clustering

The methodology discussed in this section assumes the reader is already familiar with the standard K-means objective function and algorithm described in detail in Chapter III. Additionally, for consistency and comparability, the notation used here coincides with the way the standard K-means methodology is presented in Sec. 3.2.

Instead of minimizing the standard K-means objective described Eq. 3.1, the goal of the Jacobian-Scaled (JS) clustering strategy is to minimize

$$E_{\mathbf{A}} = \sum_{k=1}^{K}\sum_{i=1}^{N}\left\|\mathbf{L}^{\mathbf{A}}_{i,k}\mathbf{A}_k[\phi_i - c_k]\right\|_2^2, \tag{5.2}$$

There are two main differences between Eq. 5.2 and the standard K-means objective in Eq. 3.1. The first difference is the presence of the weighting matrix $\mathbf{A_k} \in \mathbb{R}^{\mathbf{D}\times\mathbf{D}}$, which acts as a linear transformation, or scaling, on the standard distance vector $\delta\phi = \phi_i - c_k$. As denoted by the subscript k, this matrix is cluster-dependent through dependency on the centroid (i.e. $\mathbf{A_k} = \mathbf{A}(c_k)$). As such, the objective in Eq. 5.2 comes from a modification to the standard Euclidean distance function used to define sample similarity in the phase space. More specifically, if the Euclidean sample-centroid distance function is given as

$$d(\phi_i, c_k) = \|\phi_i - c_k\|_2^2, \tag{5.3}$$

the modified distance function implied by Eq. 5.2 is

$$d(\phi_i, c_k) = \|\mathbf{A}_k(\phi_i - c_k)\|_2^2. \tag{5.4}$$

The second difference from the standard K-means approach lies in the definition of the cluster assignment matrix, $\mathbf{L}^{\mathbf{A}}$, denoted by the superscript. The cluster assignment mechanism is based on the modified distance measure in Eq. 5.4 as

$$\mathbf{L}^{\mathbf{A}}_{i,k} = \begin{cases} 1 & \text{if } k = \underset{j}{\operatorname{argmin}} \left\|\mathbf{A}_k(\phi_i - c_k)\right\|_2^2, \text{ and} \\ 0 & \text{otherwise.} \end{cases} \tag{5.5}$$

Note that although the above cluster assignment matrix $\mathbf{L}^{\mathbf{A}}$ provides the same functionality as the standard K-means counterpart in Eq. 3.4 (i.e. it identifies the samples belonging to a particular cluster), the distance function used to prescribes the sample-centroid assignment has changed. The centroid c_k in the JS-K-means approach is defined in the same way as in the standard approach (Eq. 3.3), but instead uses this modified cluster assignment matrix:

$$c_k = \frac{\sum_{i=1}^{N} \mathbf{L}^{\mathbf{A}}_{i,k}\phi_i}{\sum_{i=1}^{N} \mathbf{L}^{\mathbf{A}}_{i,k}}. \tag{5.6}$$

The embedding of physical knowledge comes from the definition of the cluster-dependent weighting matrix $\mathbf{A}_k$. It is assumed that there exists a set of governing equations that describes the evolution of the phase space vector ϕ as

$$\frac{d\phi}{dt} = S(\phi) \in \mathbb{R}^D, \tag{5.7}$$

where the quantity $S(\phi)$ is the phase-space velocity or source term function assumed to be nonlinear. As described in Sec. 5.2, the value ϕ is the thermochemical state vector and consists of temperature and species concentrations. As such, the phase space velocity represents the chemical source term (Eq. 2.9), and the ordinary differential equation in Eq. 5.7 describes the evolution of species mass fraction and temperature in composition space. Due to the immense nonlinearity induced by the Arrhnenius-based formulation of the chemical source term (see Eq. 2.9), the above dynamical system is highly stiff and contains a very wide range of timescales. Within the context of simulations of compressible reacting flow, the above dynamical system must be solved at every simulation timestep for every cell in the computational domain to properly represent the impact of chemical reactions on the flowfield.

The method for obtaining $\mathbf{A}_k$ is now described. The leading-order Taylor expansion for the chemical source term via Eq. 5.7 is

$$S(\phi^q) - S(\phi^{ref}) = \left.\frac{\partial S(\phi)}{\partial \phi}\right|_{\phi=\phi^{ref}} \left(\phi^q - \phi^{ref}\right) + \mathcal{O}(|\delta\phi|^2), \tag{5.8}$$

where ϕ^{ref} is a reference point in composition space, ϕ^q is a query point, and $\frac{\partial S(\phi)}{\partial \phi}$ is the chemical Jacobian matrix. By setting the reference point ϕ^{ref} to the centroid c_k, the Jacobian-scaled clustering algorithm defines the expression for the scaling matrix $\mathbf{A}_k$ as the chemical Jacobian evaluated at the centroid c_k:

$$\mathbf{A}_k = \left.\frac{\partial S(\phi)}{\partial \phi}\right|_{\phi=c_k}. \tag{5.9}$$

The assumption that motivates adoption of this physics-based clustering approach is that because detonation-containing flows are fundamentally sustained by chemical reactions, the reaction contribution to the dynamics should be accounted for in the clustering and flowfield classification procedure. The idea is that one can account for this contribution by deriving the linear scaling matrix $\mathbf{A}_k$ from the governing equations in Eq. 5.7 via the chemical Jacobian. As shown below, the cluster partitions produced in this setting provide physically-consistent delineations of the flowfield – in other words, if the scaling matrix is derived from physics, the resulting clusters in phase space identify regions of increased dynamical similarity in physical space over the standard clustering algorithm, where "dynamical similarity" is defined as the degree to which two points in a local neighborhood can be described the same source term vector. Ultimately, this implies that the cluster assignments produced by the physics-based clustering approach in $\mathbf{L}^{\mathbf{A}}$ (Eq. 5.5) are more amenable for developing localized source term modeling strategies than the standard cluster labels in $\mathbf{L}$ (Eq. 3.4). This will be shown in Sec. 5.3.5.

It should be noted that by means of the Taylor expansion in Eq. 5.8 and the definition of the scaling matrix in Eq. 5.6, the distance function used in the JS-K-means approach (Eq. 5.4) can be interpreted as a linear kernel function that approximates the distance between the source terms of two points to leading order if the gradient is assumed to be locally smooth. In other words,

$$d(\phi_i, \phi_j) = \|\mathbf{A}_k(\phi_i - \phi_j)\|_2^2 \approx \|S(\phi_i) - S(\phi_j)\|_2^2 = d(S(\phi_i), S(\phi_j)). \tag{5.10}$$

As such, the objective for JS-K-means in Eq. 5.2 represents a leading-order approximation to the variation of chemical source term using only the distance between two points in composition space. If such an objective function can be minimized, the resulting clusters would identify regions of dynamical similarity by means of the assignment matrix $\mathbf{L}^{\mathbf{A}}$. Additionally, prescribing the scaling matrix as the identity (e.g.

$\mathbf{A}_k = \mathbf{I}$ for all k) recovers the baseline K-means algorithm, and implicitly assumes an identity mapping ($S(\phi) = \phi$) for the source term in the Jacobian-sacled clustering framework.

5.3.1 Differences from Other K-means Variants

The general idea of modifying the standard K-means algorithm with weighting functions is not new. In the class of density-based K-means clustering strategies, instead of modifying the distance function directly, the standard phase space distance in Eq. 5.3 is scaled with an input PDF $\rho(\phi_i)$ [146], which effectively biases the centroid locations to regions where this PDF is high. In other approaches, the distance vector $\delta\phi = \phi_i - c_k$ is scaled by a global diagonal matrix $\mathbf{D}$, producing a modified distance vector $\widetilde{\delta\phi} = \mathbf{D}\delta\phi$ that prioritizes certain components (or features) of ϕ over others in the distance evaluation, where the diagonal elements of $\mathbf{D}$ are estimated during the clustering optimization procedure [131, 338]. This amounts to a scaling operation on the phase space vector, and these approaches are typically constrained to fixing the same diagonal matrix $\mathbf{D}$ for all clusters, the advantage being that doing so leads to no required changes in the convergence properties of the baseline K-means clustering algorithm. Additionally, the class of kernel K-means strategies reformulates the distance computation by transforming the phase space variable $\phi \in \mathbb{R}^D$ into another variable $\psi \in \mathbb{R}^B$ [73, 84]. Distances are evaluated in the ψ-space by means of either the kernel trick or latent-space transformations.

A comparison of the above K-means variations with the JS-K-means approach presented here is shown in Table. 5.1. The primary differences are (a) the scaling matrix $\mathbf{A}_k$ is a function of the cluster index, which means that during an iterative optimization procedure based on the convergence of centroid locations, the $\mathbf{A}_k$ must be updated as the centroids change, (b) the scaling matrices are derived from underlying physical rules via the right-hand-side Jacobian of the dynamical system, which en-

K-means Variant	Sample-Centroid Distance	Description	Key References
Standard K-means	$\|\phi_i - c_k\|_2^2$	Standard Euclidean distance	[323]
Jacobian-scaled K-means	$\|\mathbf{A}_k[\phi_i - c_k]\|_2^2$	Cluster-dependent scaling matrix derived from dynamical system Jacobian	This work
Diagonal Weighted K-means	$\|\mathbf{D}[\phi_i - c_k]\|_2^2$	Global diagonal scaling matrix	[131]
Density-based K-means	$\rho(\phi_i)\|\phi_i - c_k\|_2^2$	Scaling of standard Euclidean distance via PDF evaluation	[146]
Kernel / Deep K-means	$\|\psi_i - c_k\|_2^2$	Standard Euclidean distance evaluated in latent space	[73, 84]

Table 5.1: Comparison of K-means variations with the Jacobian-scaled K-means approach.

sures that the clusters adapt to highly sensitive regions in the phase space, and (c) the definition of the centroid as the arithmetic mean of within-cluster samples (Eq. 5.6) is the same as the baseline algorithm, which allows for the utilization of the centroids in tabulation strategies for modeling purposes (discussed further in Sec. 5.3.3).

5.3.2 Jacobian-Scaled K-means Algorithm

The centroids are collected in the matrix $\mathbf{C} = [c_1, c_2, \dots, c_K] \in \mathbb{R}^{D \times K}$ such that the k-th column of $\mathbf{C}$ produces the corresponding D-dimensional centroid. The scaling matrices (the chemical Jacobians evaluated at the centroids) are collected in the set $\mathcal{A} = \{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_K\}$. Given these definitions, the Jacobian-scaled K-means algorithm to minimize the modified objective in Eq. 5.2 is provided in Alg. 2.

Algorithm 2 is similar to the standard K-means algorithm provided in Alg. 1 in Chapter III. The major modification is in step (b), which consists of updating the scaling matrices in $\mathcal{A}$ by computing the chemical Jacobians at the current centroid locations as per Eq. 5.9. These Jacobian matrices are required in step (c), because the labeling mechanism in JS-K-means comes from evaluating norms of the sample-centroid distance vectors scaled by the Jacobian matrix $\mathbf{A}_k$ (see Eq. 5.5). Although centroid initialization can be accomplished in the same way as with standard K-means (e.g. via the K-means++ algorithm [13]), in this work, a burn-in procedure that takes the converged centroids of the standard K-means algorithm is used for initialization of the JS-K-means centroids. Through this approach, the converged centroids produced by the JS-K-means algorithm in Alg. 2 can be interpreted as a modifier to the standard K-means centroids, which allows one to pinpoint the effect of the influence of Jacobian scaling in a clear way.

Algorithm 2 Jacobian-scaled K-means Clustering

```
Data:
(1) Set number of clusters $K$
(2) Set convergence error tolerance $\varepsilon_{tol}$
(3) Initialize dataset $\Phi = [\phi_1, \phi_2, \dots, \phi_N]$ to be clustered (Sec. 5.2).
(4) Set initial centroids $\mathbf{C}_{old}$.
(5) Initialize convergence criterion: $\varepsilon_c \leftarrow$ Inf
(6) Set value for maximum number of iterations.
Result:
(1) Converged set of centroids $\mathbf{C}_{new}$.
(2) Cluster assignment matrix $\mathbf{L}^{\mathbf{A}}$ (Eq. 5.5).
while $\varepsilon_c \geq \varepsilon_{tol}$ do
    a) Copy centroids: $\mathbf{C}_{new} \leftarrow \mathbf{C}_{old}$
    b) Evaluate Jacobians (scaling matrices) at $\mathbf{C}_{old}$: $\mathcal{A} \leftarrow Eq.$ 5.9
    c) Update cluster assignment matrix using $\mathcal{A}$ and $\mathbf{C}_{old}$: $\mathbf{L}^{\mathbf{A}} \leftarrow Eq.$ 5.5
    d) Update centroids using cluster assignment matrix: $\mathbf{C}_{new} \leftarrow Eq.$ 5.6
    e) Compute convergence criterion: $\varepsilon_c \leftarrow \|\mathbf{C}_{new} - \mathbf{C}_{old}\|_F$
    f) Break if maximum number of iterations is reached
end
```

Note that in Alg. 2, the definition of the centroid during the iterative convergence procedure has not changed from the standard K-means approach, but the assignment mechanism has changed by means of a different distance function. This produces an inconsistency, because the way in which the centroid update rule is derived is intrinsically tied to the distance function used in the definition of the K-means objective. Since the distance function has changed in the Jacobian-scaled approach, and the definition of the centroid has not changed, this inconsistency translates to eliminating the convergence guarantees of the standard K-means algorithm.

More specifically, at a given iteration, the updated centroids are contained $\mathbf{C}_{new}$ and the previous centroids are contained in $\mathbf{C}_{old}$. In the standard K-means approach, the goal is to derive an update rule for $\mathbf{C}_{new}$ based on the cluster labels $\mathbf{L}^{\phi}$ generated from the previous centroids $\mathbf{C}_{old}$ that results in a minimization of the objective function in Eq. 3.1. As shown in *Bottou and Bengio* [37], the update rule for the K centroids is Eq. 3.3.

However, in JS-K-means, the objective function is different from standard K-means (Eq. 5.2 instead of Eq. 3.1). A derivation of the update rule for $\mathbf{C}_{new}$ that minimizes this new objective yields a slightly different expression. For centroid $c_{k,new}$, this update rule is

$$c_{k,new} = \frac{\sum_{i=1}^{N} \mathbf{L}^{\mathbf{A}}_{i,k} \phi_i}{\sum_{i=1}^{N} \mathbf{L}^{\mathbf{A}}_{i,k}} + R. \tag{5.11}$$

In the above equation, the true centroid update for minimizing the physics-based K-means objective in Eq. 5.2 departs from the one used in the algorithm by a residual R. Described detail in Appendix A, the residual is proportional to both the within cluster variance and the rate-of-change of the matrix $\mathbf{A}_k$ in the D-dimensional phase space (e.g. the chemical Hessian matrix). As shown in Sec. 5.3.5, ignoring the residual in Eq. 5.11 and retaining the original centroid definition ensures that the clusters are localized in phase space at the cost of guaranteeing a monotonic decrease of the modified objective function in Eq. 5.2 during the iterative procedure. In practice, good convergence trends are still observed via Alg. 2, although future work involves replacing the standard centroid update with the "true" update defined by Eq. 5.11. The reader is refferred to Appendix A for more information on derivations for the centroid update rules.

5.3.3 Localized Source Term Modeling

The expression in Eq. 5.8 facilitates a localized source term modeling framework based on a linear tabulation strategy. If both the chemical source term and chemical Jacobians are stored at the K centroids (Jacobians are already required as a part of the modified K-means algorithm), estimates of the chemical source term at a given query point in composition space ϕ^q can be obtained as

$$\mathbf{S}(\phi^q) \approx \mathbf{S}(c_k) + \mathbf{A}_k \left(\phi^q - c_k\right), \tag{5.12}$$

for ϕ^q residing in cluster k. Recall that the idea of utilizing K-means clusters in composition space for localized source term modeling was also explored in Chapter IV with the training of ANNs. The linear evaluation approach in Eq. 5.12, however, is advantageous due to its simplicity and inherent compatibility with the JS-K-means strategy. If acceptable convergence in the centroids is observed, the clusters in the Jacobian-scaled approach are expected to be biased towards regions in composition space that are directly compatible with the linear prediction framework in Eq. 5.12. Put another way, because the goal of the physics guided clustering framework is to partition the D-dimensional composition space into regions that are similar in source term, recovery of the source term via Eq. 5.12 becomes feasible, as will be shown in Sec. 5.3.5.

Although the linear estimation is less expressive than ANN-based approaches, it provides two main advantages: (1) elimination of the need for a training stage, and (2) interpretability and control of error. The second quality is particularly appealing and is the object of Sec. 5.4 – based on an error estimator derived from Eq. 5.12, Sec. 5.4 shows how clusters producing high errors can be identified and refined such that a preset tolerance is satisfied, leading to accurate source term estimates and, most importantly, elimination of K as a parameter in the clustering procedure.

Before proceeding, it should be noted that the prediction approach of Eq. 5.12 based on linear extrapolations about reference points (which here are centroids) in phase space is similar in spirit to the methodology of in-situ adaptive tabulation (ISAT) developed by *Pope* [263]. Although the linear predictive framework is the same, the tabulation and querying strategies are fundamentally different. In ISAT, the partitioning mechanism is a set of ellipsoids produced by singular value decompositions (SVDs) of chemical Jacobian matrices at reference points which coincide with points in the input dataset. Source terms for query points within the ellipsoids are extracted based on extrapolation from the ellipsoid centers. The querying mechanism consists of projections onto hyperplanes in the subspace spanned by the singular vectors to assign the point to an ”owner” ellipsoid [263]. In the physics-guided clustering approaches presented here, the partitioning is a space-filling tessellation of composition space – as such, the querying procedure is much simpler due to (a) cluster assignment mechanisms by means of nearest-centroid distance evaluations (Eq. 5.5), and (b) ensuring no ambiguity with regards to which partition (or cluster) the query point belongs. Further, the reliance on SVD-facilitated projection is removed, which is a benefit in situations where this projection is ill-defined (e.g. in cases of non-reacting species or chemical equilibrium). Lastly, the K-means approach presented here produces a tessellation of composition space that seeks to minimize a global objective function (Eq. 5.2) that by design ensures similarity in source terms within each cluster, which is amenable for the modeling task in Eq. 5.12. It is acknowledged, however, the the ISAT framework is much more developed than the strategy presented here, and is more directly compatible and deployable for in-situ source term modeling in combustion solvers – as mentioned in Chap. VI, direct comparison of source term predictions bewteen the physics-guided clustering strategies developed here and ISAT is an object of future work.

5.3.4 Scaling Procedure for Jacobian Regularization

A scaling procedure is carried out before clustering such that (a) the individual components of ϕ (species concentration and temperature) have equal importance during the clustering procedure, and (b) the Jacobian matrices are regularized. The scaled phase space variable, denoted $\widetilde{\phi}$, is recovered from the original phase space variable ϕ_i via

$$\widetilde{\phi} = \mathbf{B}\phi = \begin{bmatrix} \frac{1}{\max(\phi^1)-\min(\phi^1)} & & \\ & \ddots & \\ & & \frac{1}{\max(\phi^D)-\min(\phi^D)} \end{bmatrix} \begin{bmatrix} \phi^1 \\ \vdots \\ \phi^D \end{bmatrix}. \tag{5.13}$$

In the above equation, superscripts denote component indices (not powers) and $\mathbf{B}$ is a diagonal scaling matrix that normalizes each respective component of ϕ by its range. An analogous scaling matrix can be applied for the ground-truth source term data as

$$\widetilde{S(\phi)} = \mathbf{C}S(\phi) = \begin{bmatrix} \frac{1}{\max(S(\phi)^1)-\min(S(\phi)^1)} & & \\ & \ddots & \\ & & \frac{1}{\max(S(\phi)^D)-\min(S(\phi)^D)} \end{bmatrix} \begin{bmatrix} S(\phi)^1 \\ \vdots \\ S(\phi)^D \end{bmatrix}. \tag{5.14}$$

Given the diagonal scaling matrices for the composition variable and source term ($\mathbf{B}$ and $\mathbf{C}$ respectively), the scaled Jacobian matrix, denoted $\widetilde{\mathbf{A}}$, can recovered as

$$\widetilde{\mathbf{A}} = \mathbf{D} \odot \mathbf{A}, \quad \mathbf{D} = \text{diag}(\mathbf{C})\text{diag}(\mathbf{B}^{-1})^{\mathrm{T}}, \tag{5.15}$$

where $\odot$ denotes an elementwise matrix product (Hadamard product) and the diag$(\cdot)$ operation produces a column vector containing the diagonals of the input matrix. The disadvantage here is that obtaining the Jacobian scaling matrix $\mathbf{D}$ requires knowledge of the ranges (minimum and maximim values) of both composition and chemical source term elements, which in practice are extracted from the input dataset – the consequence is that the scaling matrices will be application or problem dependent. Expert guided knowledge can be used to obtain ranges for composition values (e.g. temperatures in detonations typically fall within the range of 200K to 6000K), but the ranges for source term values cannot be prescribed this way. The advantage in the scaling procedure, however, is that it non-dimensionalizes all variables involved in the Taylor expansion (Eq. 5.8), which leads to improved predictive capability for minor species concentrations via Eq. 5.12 as well as improved convergence of the JS-K-means procedure in Alg. 2. This is evidenced in Fig. 5.3, which compares the singular value distributions of the scaled and unscaled chemical Jacobians obtained from the detonation dataset Φ. The scaling procedure can therefore be interpreted as a type of preconditioner for the Jacobians, as it directly drops the disparity between the largest and smallest singular values in the data distribution to be used in the clustering procedure.

5.3.5 Results

5.3.5.1 Toy Problem

Before proceeding with the detonation dataset, this section demonstrates the JS-K-means algorithm on a simple 1D toy problem. The intent is to illuminate the manner in which inclusion of the Jacobian-scaled distances in the iterative procedure in Alg. 2 modifies the centroid locations (and resultant cluster labels) produced by the standard K-means procedure. In this toy problem, the underlying nonlinear

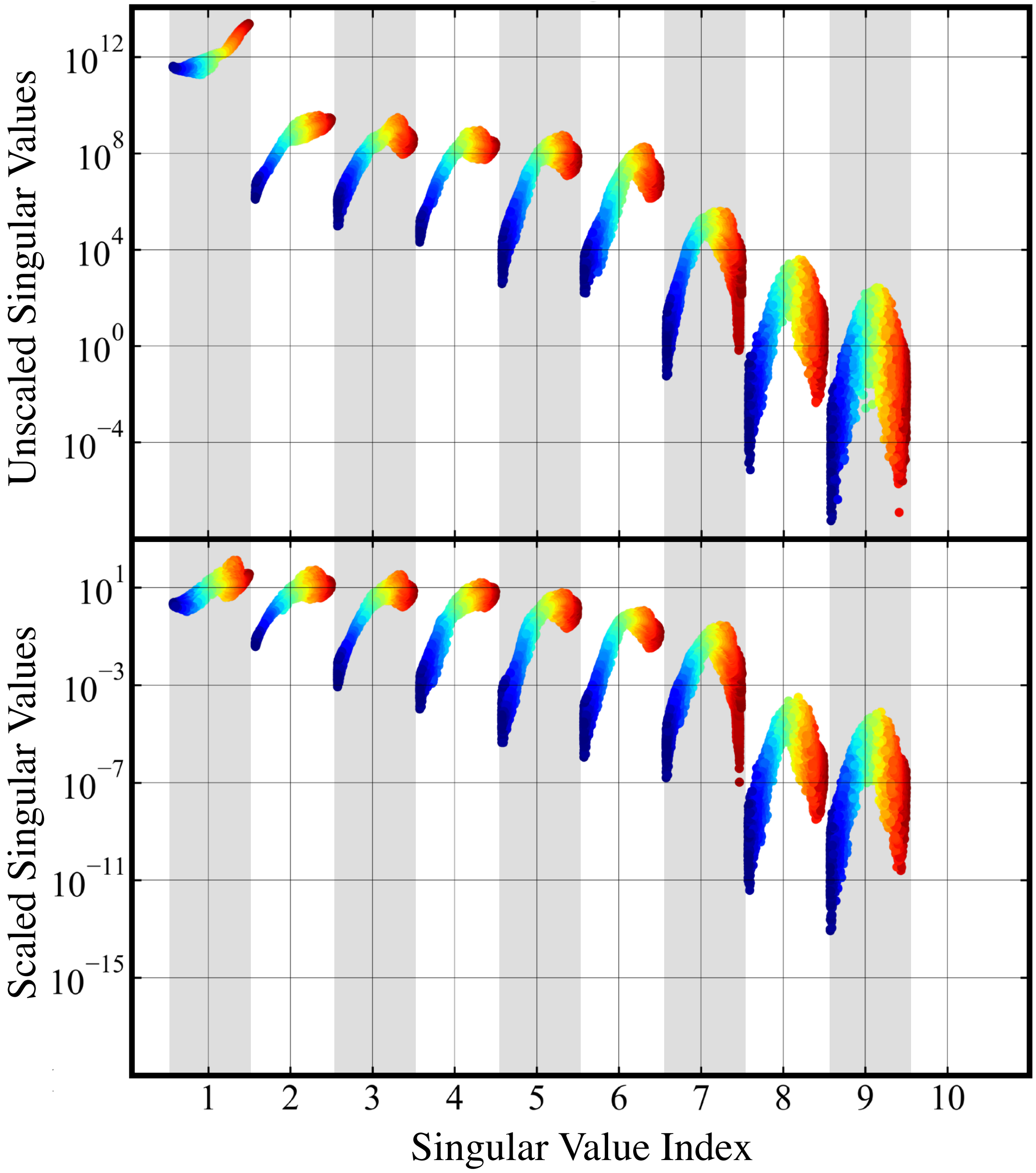


Figure 5.3: Singular value distribution for the dataset Φ (see Sec. 5.2) derived from unscaled (top) and scaled (top) chemical Jacobians. The x-axis denotes the singular value index. For each index, spread in singular value for all N sample points is plotted and colored by temperature – 10th index is undefined due to presence inert N2 species. Bands are alternately shaded for ease of visualization.

dynamical system is represented as a quadratic function via

$$\frac{d\phi}{dt} = S(\phi) = \frac{1}{2}\phi^2 \in \mathbb{R}, \tag{5.16}$$

which produces the trivial Jacobian $\partial S/\partial\phi = \phi$. The phase space variable ϕ is sampled within the range $[0, 10]$ at $N = 10000$ evenly spaced intervals. A plot of the quadratic source term and linear Jacobian from Eq. 5.16 is shown in Fig. 5.4 – note that the variables in Fig. 5.4 have been scaled as per the procedure outlined in Sec. 5.3.4. The figure also provides convergence trends for both standard K-means (Eq. 3.1) and JS-K-means procedures (Eq. 5.2) resulting from a number of clusters $K = 10$. The convergence diagram follows the burn-in procedure discussed in Sec. 5.3.2: the standard K-means algorithm is run for a prescribed number of iterations (here 300) using centroids initialized from the K-means++ procedure [13], upon which the converged centroids from the standard procedure are fed into the modified algorithm for the same number of iterations (or until the convergence criterion is met). Convergence trends in Fig. 5.4 are visualized by plotting objective functions normalized by their maximum observed value for both the standard and Jacobian-scaled approach throughout these iterations.

During the burn-in procedure (which utilizes the standard K-means algorithm), the standard objective function which encapsulates the within-cluster sum of squares decreases monotonically as expected – interestingly, the JS-K-means objective function also decreases in a similar monotonic behavior, despite the fact that the standard objective is being minimized. At iteration 300, the converged centroids from the standard approach are then supplied as the initial centroids to the Jacobian-scaled approach. The effect on both objective functions after this switch is in-line with the goal of the JS-K-means algorithm. Upon switching to the JS-K-means algorithm, the objective function – which represents an approximation to the true within-cluster

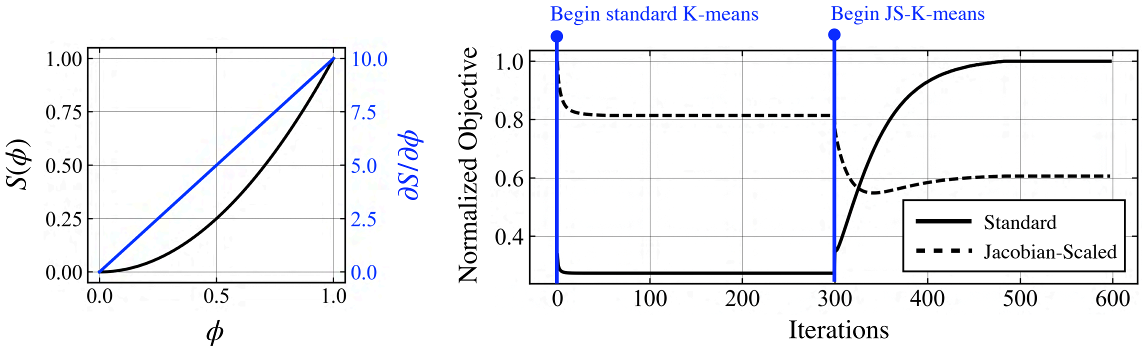


Figure 5.4: **(Left)** Plot of source term (black) and Jacobian (blue) versus phase space variable ϕ for the 1D toy problem in Eq. 5.16. **(Right)** Plot of normalized objective functions for standard (solid) and physics-guided (dashed) K-means clustering approaches versus number of iterations. The first 300 iterations is a burn-in phase that utilizes the standard K-means algorithm in Alg. 1. The next 300 iterations utilize the modified physics-guided K-means algorithm in Alg. 2.

variation in source terms using distances in phase space scaled by the Jacobians evaluated at the centroids – experiences a significant drop, and the standard objective experiences a rise. This is indicative of nonlinearity in the source term, as it implies that optimizing the cluster partitions in phase space for source term variation necessitates a tradeoff in the optimization of the clusters for variation in the state variable. Additionally, as described in Sec. 5.3.2, the JS-K-means algorithm does not guarantee monotonic decrease of the corresponding objective function; instead, convergence of Alg. 2 is indicated by a stabilization of the objective function curves at an equilibrium point for which the centroids are stabilized in the D-dimensional phase space.

Ultimately, Fig. 5.4 shows how the Jacobian-scaled K-means algorithm drops the corresponding Jacobian-based objective function in Eq. 5.2 as intended, producing a set of converged centroids that attempt to minimize variation in nonlinear source terms within the clusters. To better interpret this effect, a visualization of the change in centroid locations provided by the JS-K-means algorithm with reference to the converged centroids from the standard K-means algorithm is provided in Fig. 5.5 for $K = 5$, 10, and 15 clusters (note that, although not shown, the convergence trends in Fig. 5.4 provided for $K = 10$ hold for the other values of K). In all cases,

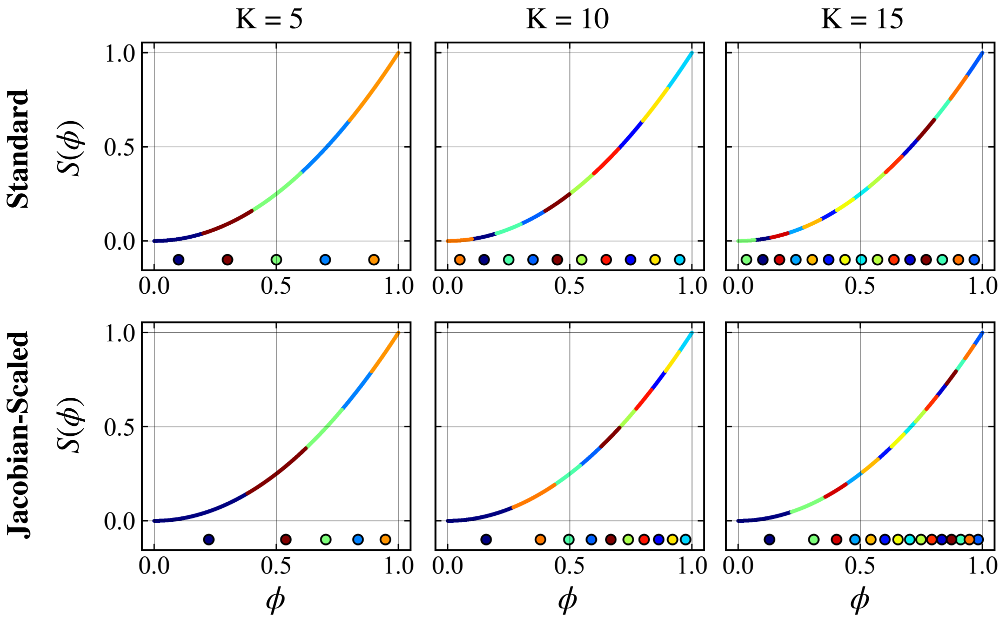


Figure 5.5: Cluster visualizations provided by the standard K-means algorithm (top row) and physics-guided K-means algorithm (bottom row) for $K = 5$, 10, and 15. Centroid locations are provided as the filled markers in each plot.

Fig. 5.4 shows how the physics-guided algorithm shifts and redistributes the standard K-means clusters towards high-sensitivity regions in the phase space. This effect can be quantitatively accessed by correlating the cluster sizes with centroid Jacobian norms, as shown in Fig. 5.6 – the weighting towards high-sensitivity regions implies that clusters are redistributed in regions where there is significant nonlinearity. Put another way, for a finite set of K centroids, the physics-guided clustering algorithm through Jacobian-scaling ensures that the K centroids are localized in regions of dynamical similarity as prescribed by the governing equations via Eq. 5.16.

5.3.5.2 Detonation Dataset: Analysis of Clusters

The JS-K-means clustering algorithm is now extended to the detonation dataset described in Sec 5.2. More specifically, in a similar manner as in the toy problem above, convergence trends for various K values will be analyzed, and changes in cluster assignments in both physical space and phase space due to Jacobian-scaling

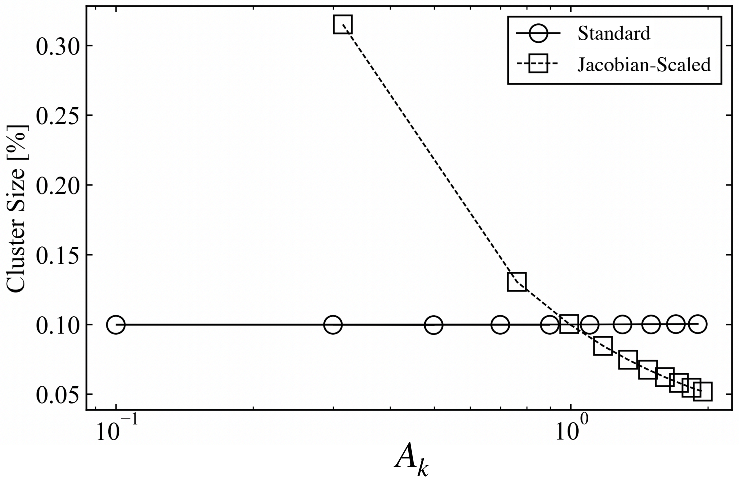


Figure 5.6: Cluster size versus Jacobian value evaluated at centroid A_k for standard K-means and physics-guided K-means approaches in the 1D toy problem ($K = 10$). Cluster size is defined as percentage recovered from the ratio of within-cluster samples to total number of samples.

will be interpreted. Additionally, analysis of the linear source term prediction outputs via Eq. 5.12 using phase space partitions produced by both standard and Jacobian-scaled approaches is performed.

Before visualization and interpretation of the partitioned composition space, it is important to ensure that the algorithm presented in Alg. 2 produces the a similar decrease in the physics-based loss in the detonation dataset as observed in the 1d toy problem in Fig. 5.4. To this end, convergence trends for both standard (Eq. 3.1) and Jacobian-scaled (Eq. 5.2) objective functions are provided in Fig. 5.7 for $K = 5$, 15, and 30. For each cluster number, the various curves (10 for each K value) represent a different set of initial centroids provided to the standard K-means algorithm to initiate the burn-in procedure, resulting in unique K-means runs. Because the K-means algorithm depends on centroid initialization, analysis of performance should take into consideration the effect of stochasticity due to centroid initialization.

Overall, the convergence trends in Fig. 5.7 show how Alg. 2 drops the physics-based Jacobian-scaled objective in Eq. 5.5 even for the complex detonation dataset. As with the toy problem, there is no noticeable oscillatory behavior in the objective curves, indicating that the centroid locations have stabilized. Additionally, the effect of K-means stochasticity due to initial centroid variation, as visualized by the spread in objective function curves, is lowest for $K = 5$ and highest for $K = 30$ as expected. Despite this, in all cases, the spread due to different centroid initialization in all cases is minimal and inconsequential in terms of converged objective functions. The trends in Fig. 5.7 imply that the advantage provided by the JS-K-means algorithm in dropping the Jacobian-scaled objective function (Eq. 5.2) increases with cluster number. In other words, the drop-off in the Jacobian-scaled objective provided by the modified algorithm in Alg. 2, relative to the same objective function evaluated with centroids produced by the standard K-means algorithm, increases with higher cluster numbers. This is a motivating result with regards to scalability in terms of K.

Figure. 5.8 shows the effect of cluster number on the objective function in Eq. 5.2 computed from both converged standard K-means clusters and JS-K-means clusters. For consistent comparison across cluster numbers, a normalized version of the physics-based objective, termed the root-mean-square (RMS) objective, is plotted instead of the value in Eq. 5.2. This RMS objective is given by

$$E_{\mathbf{A}}^{RMS} = \sqrt{\frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{N}\left\|\mathbf{L}_{i,k}^{\mathbf{A}}\mathbf{A}_k[\phi_i - c_k]\right\|_2^2}. \tag{5.17}$$

Recall that the initial set of centroids provided as input to the JS-K-means algorithm come from the converged standard K-means centroids. As such, to produce the curves in Fig. 5.8, the above RMS objective is computed using centroid and distance evaluations from both standard and Jacobian-scaled approaches. The trends in Fig. 5.8 show how the modification of centroid locations provided by the JS-K-means approach results in decreased within-cluster variations of source term. An increasing cluster number in both standard and JS-K-means approaches results in a nearly

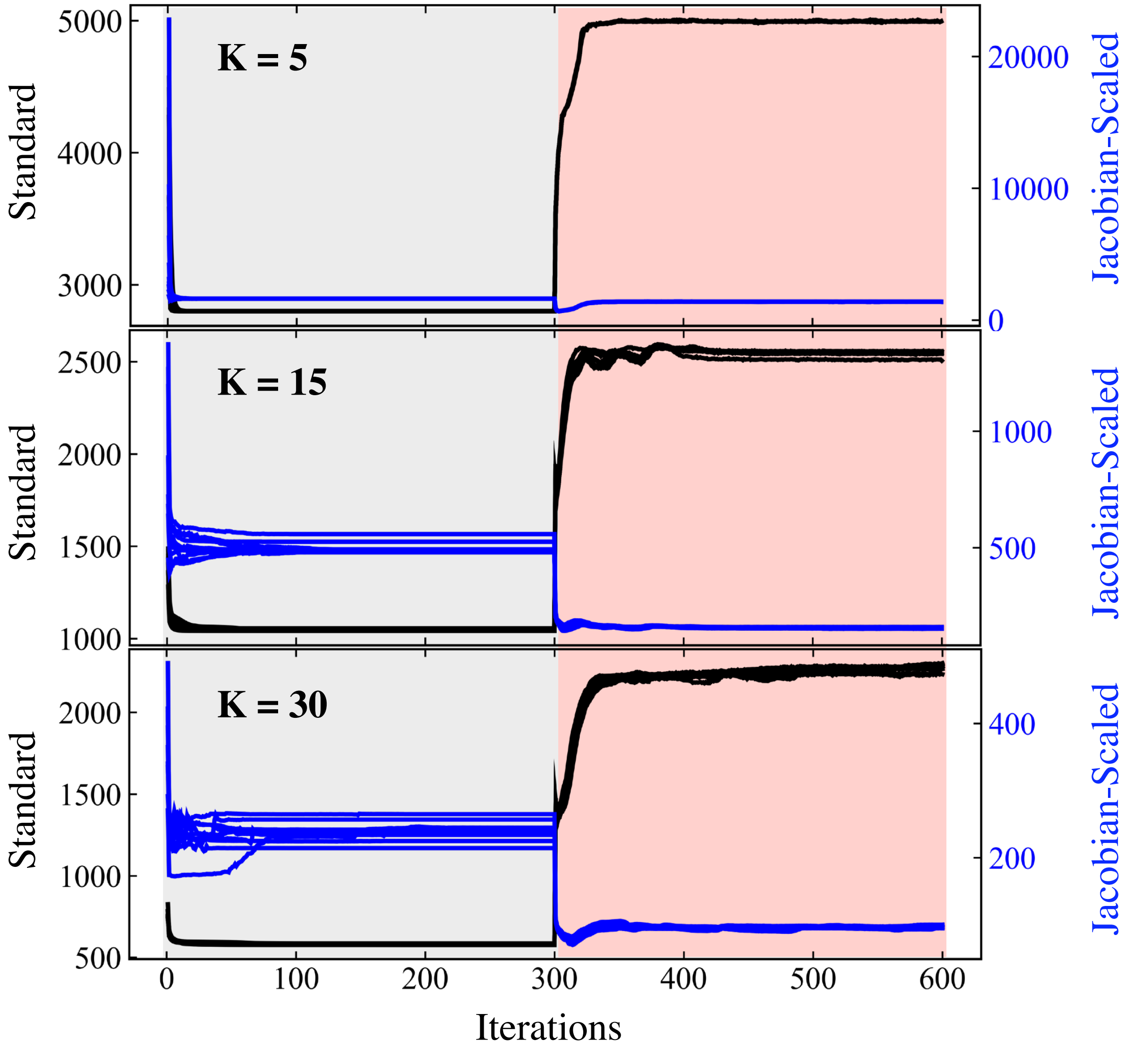


Figure 5.7: Standard K-means (black, Eq. 3.1) and physics-based K-means (blue, Eq. 5.2) objective function values during the iterative procedure for $K = 5$ (top), $K = 15$ (middle), and $K = 30$ (bottom). As in Fig. 5.4, the gray shaded regions denote the burn-in period in which the standard K-means algorithm is run for 300 iterations – the red shaded region denotes the switch to the JS-K-means algorithm in Alg. 2.

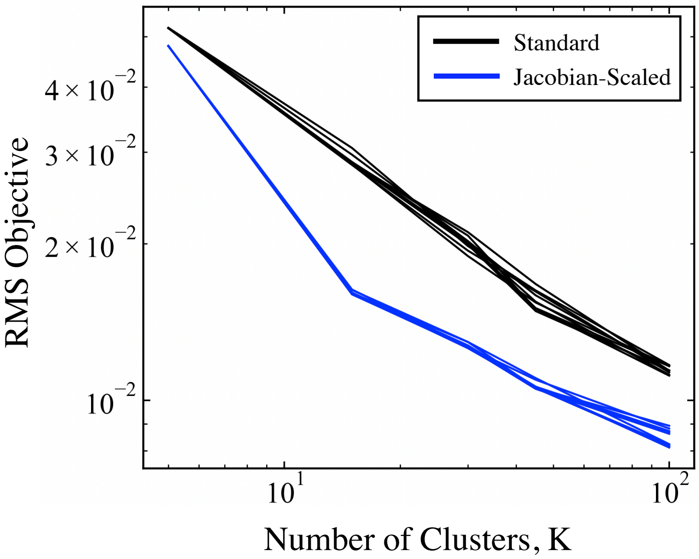


Figure 5.8: Evaluation of physics-based RMS objective function (Eq. 5.17) versus number of clusters using standard K-means output (black) and JS-K-means output (blue).

constant-rate reduction in the Jacobian-based RMS objective, which is indicative of the localization of clusters. Interestingly, despite the fact that the standard K-means approach does not optimize Eq. 5.17, higher cluster numbers still result in lower values of Eq. 5.17, implying that source terms are localized in composition space for this dataset. Usefully, even at higher cluster numbers (e.g. $K = 100$), the gap in RMS objective between the JS-K-means and standard K-means evaluations remains, which signifies that the JS-K-means algorithm succeeds in shifting the baseline clusters produced by standard K-means towards regions of greater dynamical similarity.

Although informative, analysis of trends in the objective function values must be supplemented with a visual inspection of the cluster partitions in physical space for a clearer interpretation of the impact of the JS-K-means clustering procedure. To this end, cluster assignments obtained from standard K-means and JS-K-means are shown in Fig. 5.9 for the $K = 15$ case – general trends discussed hereafter apply to all studied K values. Note that the actual colors used to facilitate visualization of the cluster labels in physical space come from the cluster index, and are not meaningful from a physical perspective – however, as with Chapter IV, coherent regions in physical space defined by cells of the same color (or cluster) are indicators for key flow features extracted by the unsupervised algorithms. Borrowing terminology from Chapter IV, the cluster label plots in physical space are termed the standard (obtained from standard K-means) and Jacobian-scaled (obtained from JS-K-means) segmented fields respectively.

The segmented fields in Fig. 5.9 reveal the way in which the JS-K-means algorithm biases, or pushes, the cluster labels towards the wavefront and detonation reaction zone (indicated by the white arrows in the repsective plot). This is consistent with the fact that the difference between the standard and Jacobian-scaled approach is in the chemical Jacobian based scaling of the distance function, which necessitates the localization (or redistribution) of centroids towards regions of high composition sensitivity. This phenomenon is directly analogous to the biasing and weighting of the centroids seen in the toy problem in Fig. 5.5. As a result, the redistribution of centroids in composition space provided by the JS-K-means algorithm effectively translates to localizing the cluster partitions near regions of high chemical stiffness, which is a useful property for modeling purposes that intend to target and eliminate stiffness from the simulation procedure. On the other hand, the standard K-means approach produces a segmented field that is significantly more refined and complex in turbulence-dominated regions far behind the wavefront that are unimportant from the chemical contribution perspective.

For a better visualization of the algorithm behavior near the wavefront, Fig. 5.10 shows a series of zoomed-in profiles of pressure, density, heat release rate, and the same segmented fields from Fig. 5.9. Before assessing the near-wavefront trends in the segmented fields, it should be noted that the pressure profile shown in Fig. 5.10 displays the characteristic triple-point structures that oscillate vertically along the wavefront as it propagates through the channel. In short, triple points are local-

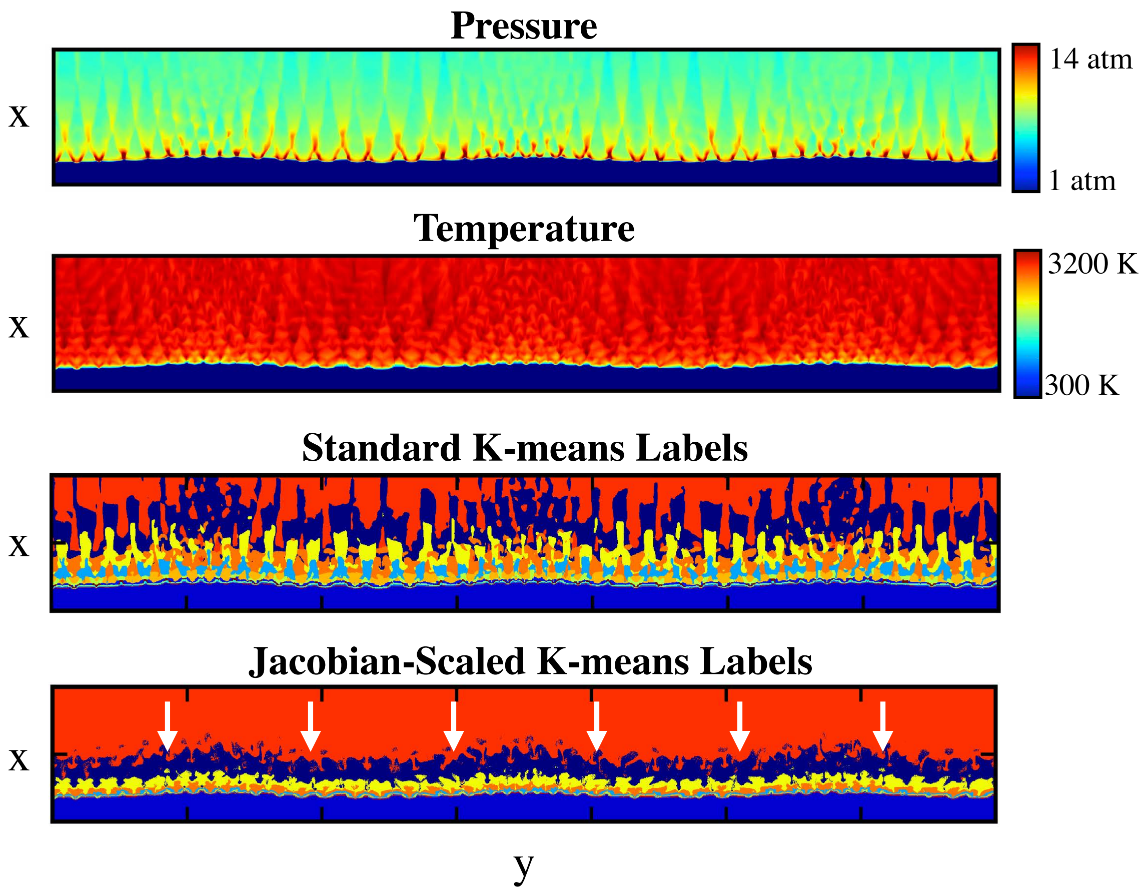


Figure 5.9: From top-to-bottom: pressure, temperature, standard K-means labels, and JS-K-means labels for detonation dataset as described in Sec. 5.2. Flowfields have been transposed (wave is moving towards bottom of page) for ease of visualization.

ized high-pressure regions at the detonation wave front emanating from collisions between a weaker series of reflecting transverse waves and the leading shock wave which travels at the CJ speed [311]. The triple point structures, indicated by the small distributed regions of peak pressure throughout the wavefront, contribute to significant complexity in the detonation wave dynamics and are known to heavily influence chemical kinetic behavior within the detonation wave structure [278]. As such, part of assessing the strength of any composition space partitioning algorithm for detonation-containing flows is to ensure that areas near and within the triple point structures are detected by the given algorithm with ideally minimal user input.

The differences between the segmented fields in this triple point region are quite

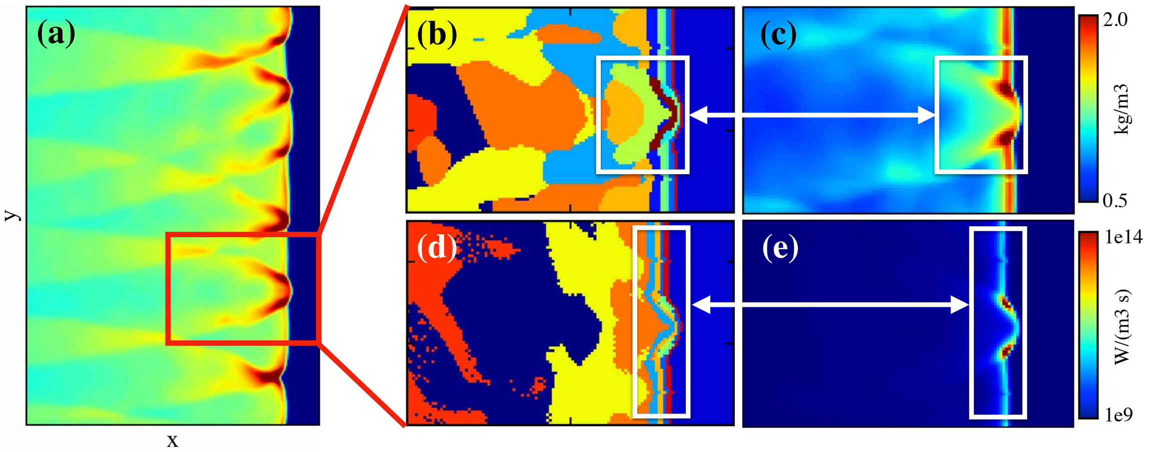


Figure 5.10: **(a)** Pressure profile within detonation wave in the domain window of $x = [0.087, 0.094]$ m and $y = [0.04, 0.05]$ m (coordinate axes supplied in Fig. 5.2). Red box indicates zoom-in region on triple point structures for remainder of plots in the figure. **(b)** Standard K-means labels in triple point region. **(c)** JS-K-means labels in triple point region. **(d)** Fluid density (kg/m^3) in the triple point region. **(e)** Heat release rate (W/m^3/s) in the triple point region. White boxes in (b)-(e) indicate correspondence in respective segmented field structure and key flow features.

apparent in the zoom-ins – for example, in the standard K-means segmented field, the cluster partitions attempt to recover spatially coherent patterns that align more with density fields and contact discontinuities away from the wavefront, which contribute to the complex and somewhat turbulent cluster partitions observed downstream. This comes from the definition of the composition vector, which consists of temperature as well as density-weighted mass fractions (i.e. species concentration); as such, there is an intrinsically higher weight placed on non-reacting transverse wave propagation in the standard clustering procedure, which in turn leads to complex 2-dimensional structures recovered in the segmented field.

On the other hand, the segmented fields produced by the JS-K-means approach are markedly more 1-dimensional in overall structure (e.g. the variation in cluster transitions along the y-direction is noticeably smaller than that observed in the standard K-means segmented fields). The effect of biasing clusters towards the wave front, which was visualized macroscopically in Fig. 5.9, is also apparent in the zoom-ins. The JS-K-means algorithm pushes clusters towards the regions of high chemical reactivity/sensitivity, as encoded in corresponding regions of peak heat release rate and the triple points. Because the contact discontinuities and transverse wave oscillations behind the detonation wavefront produce much less chemical reactivity as implied by the heat release rates, the cluster partitions from the Jacobian-scaled approach are more vertically uniform behind the wavefront. Additionally, there is a higher degree of noise in the physics-based segmented field than in the standard counterpart – this is due to potentially high degrees of chemical stiffness, as well as possible numerical truncation errors attributed to numerical chemical Jacobian estimates for intermediary species.

The above discussion is geared towards interpretation of the output of the JS-K-means clustering procedure in relation to the standard K-means approach in physical space. However, additional insight into the workings of the JS approach can be extracted by visualizing the cluster distributions in composition space. Because the composition space is high-dimensional ($D = 10$), direct visual analysis of cluster partitions is not straightforward. To facilitate an interpretable composition space analysis, the approach here utilizes projections onto two-dimensional spaces facilitated by proper orthogonal decomposition (POD). It is assumed that the reader is already familiar with the basics of POD – fundamentals and background on the approach are provided in Sec. 2.5.2 and the references therein.

To better visualize the effects of JS-K-means, two sets of POD bases are derived: one from the dataset containing the phase space variables Φ, and another from the dataset containing the ground-truth source terms Ω. The POD basis derived from Φ is denoted $\mathbf{U}_\Phi \in \mathbb{R}^{D \times M}$, and the POD basis derived from Ω is denoted $\mathbf{U}_\Omega \in \mathbb{R}^{D \times M}$. The columns of the matrices $\mathbf{U}_\Phi$ and $\mathbf{U}_\Omega$ contain the respective basis vectors (also known as principal component directions), and M denotes the number of retained modes in the expansion. Projection of the datasets onto their respective POD basis vectors produces the POD coefficients, the components of which are uncorrelated due to the orthonormal property of the basis vectors. More formally, the phase space coefficients are obtained from the projection

$$\mathbf{A} = [a_1, \dots, a_N] = \mathbf{U}_\Phi^{\mathrm{T}} \Phi \in \mathbb{R}^{M \times N}, \tag{5.18}$$

and the source term coefficients from the analogous projection

$$\mathbf{B} = [b_1, \dots, b_N] = \mathbf{U}_\Omega^{\mathrm{T}} \Omega \in \mathbb{R}^{M \times N}. \tag{5.19}$$

In the above equations, the a_i and b_i are M-dimensional column vectors for the i-th phase space and source term coefficient, respectively. One of the useful properties of POD is that if the number of modes M is prescribed as the maximum (e.g. if M is equal to the rank of the corresponding data matrices), the Euclidean distance between two points in the phase space is equivalent to the Euclidean distance between the same two points in the projected space defined by the POD basis. In other words, for two samples i and j, if all basis vectors are retained in the POD representation, the distance between two points ϕ_i and ϕ_j in composition space can be cast as

$$\|\phi_i - \phi_j\|_2^2 = \|a_i - a_j\|_2^2, \tag{5.20}$$

and similarly, the distance between the same two points in a so-called "source-term" space can be cast as

$$\|S(\phi_i) - S(\phi_j)\|_2^2 = \|b_i - b_j\|_2^2. \tag{5.21}$$

The above property comes from the fact that the data variance is preserved by design in the POD expansion if all modes are retained [29]. Mode "energies" extracted from the eigenvalues of respective covariance matrices can then be used to quantify the percentage amount that each POD mode contributes to the overall data variance. This is shown in Fig. 5.11 – usefully, it is shown how for both composition and source

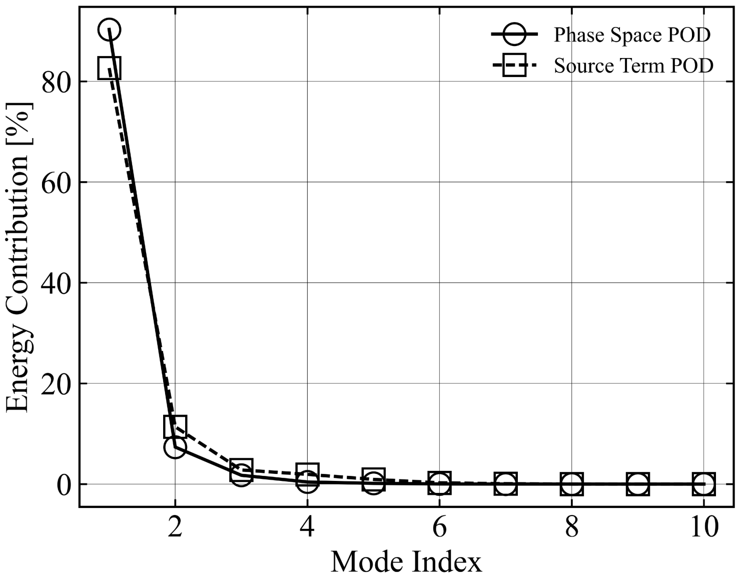


Figure 5.11: Energy contribution versus mode index, measured as a percentage of total variance captured by the POD modes, from the phase/composition space decomposition (Eq. 5.18) and source term decomposition (Eq. 5.19).

term data projections, the first two POD modes retain a majority of the data variance. As such, 2-dimensional visualizations utilizing only the first two POD components from both the composition dataset (Eq. 5.18) and source term dataset (Eq. 5.19) can be used to directly assess the effects of the JS-K-means clustering procedure in terms of centroid distributions and cluster labels.

Figure 5.12 shows examples of these visualizations in both the phase space coefficients (a_i, Eq. 5.18) and source term coefficients (b_i, Eq. 5.19) for select composition variables and their corresponding source terms. There is significant correlation between the POD coefficients and key quantities of interest such as temperature fields, H2O mass fraction (which can be interpreted as a reaction progress variable), as well as chemical source terms; these correlations confirm that the respective POD projections facilitate a concise representation of data variance in both composition and source term space in two dimensions. Note that in the source term space (bottom row of Fig. 5.12), the regions in composition space (top row) that produce zero chemical source term collapse to a single point (the $(0, 0)$ coordinate). Usefully, the correla-

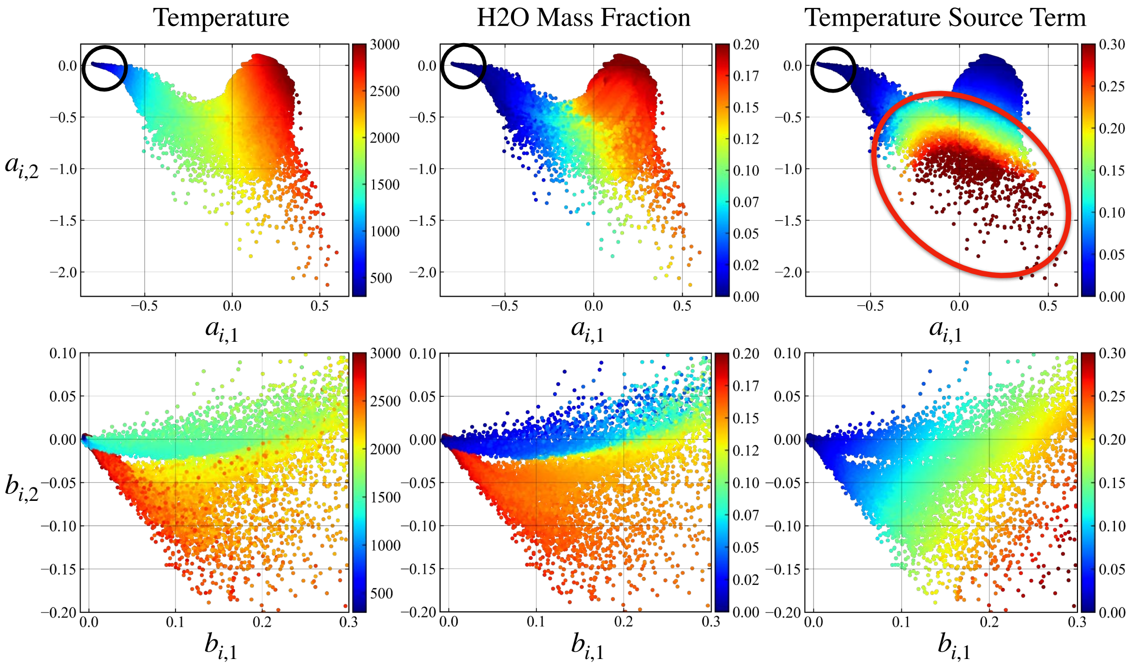


Figure 5.12: **(Top row)** From left-to-right, visualization of temperature field in units of Kelvin, H2O mass fraction, and temperature source term (nondimensinoalized as per Sec. 5.3.4) in the two-dimensional composition POD coordinates (Eq. 5.18). Black circle indicates ambient region (unreacted gas ahead of the detonation wave), and red circle indicates regions of high chemical heat release at reactivity (high-sensitivity regions within the detonation wave structure). **(Bottom row)** Same as top row, but for source term POD coordinates (Eq. 5.19).

tion in temperature source term becomes much more linear in the source term space as opposed to the composition space – in other words, an increase in values in the first principle component of the source term POD coefficient produces, to reasonable confidence, an increase in chemical reactivity.

Figure 5.13 compares the centroid locations and corresponding cluster labels produced by the standard K-means with those produced by JS-K-means in the two-dimensional composition POD space. Figure 5.14 displays the same quantities in the source term POD space. The plots in Figs. 5.13 and 5.14 illustrate (a) the way in which the JS-K-means modifies centroid locations from the standard K-means output at a given K (e.g. the cluster biasing or shifting effect), and (b) how JS-K-means refines clusters in fundamentally different regions of phase space than in the standard approach. More specifically, in Fig. 5.13 for a prescribed number of clusters, the JS-K-means algorithm successfully redistributes centroids away from the ambient, non-reacting regions (black circles) and towards the highly reactive regions (red circles) that characterize much of the complexity in the detonation wave structure (i.e. regions of peak heat release rates). With regards to cluster refinement, the physics-guided approach of JS-K-means refines clusters in this same reactive region, which is equivalent to allocating additional clusters near the detonation wavefront and triple point structures, where high chemical stiffness and species sensitvity is expected to occur. In the same effect, increasing the number of clusters in the Jacobian-scaled approach does not result in a refinement of the ambient, non-reacting region of the flowfield, which is not true for standard K-means. This effect of centroid redistribution is particularly apparent in the source term POD space projections of Fig. 5.14, which shows how JS-K-means increases the degree of spread of centroids in source term space, which in turn drops the within-cluster variation of source term distances within each cluster. This not only supports the segmented field behavior of the JS-K-means clustering approach in physical space (Figs. 5.9 and 5.10), but also confirms the fact that the modified algorithm in Alg. 2 is properly minimizing the modified objective in Eq. 5.2.

5.3.5.3 Detonation Dataset: Source Term Predictions

The advantage of JS-K-means is that centroids are pushed towards regions in which source terms are similar without invoking source term evaluations on the entire dataset – this is useful for applying the JS-K-means partition to produce *a-posteriori* source term estimates in solvers for combustion modeling. To this end, as a first step, the discussion below assesses *a-priori* localized source term estimates using the linear approximation method outlined in Sec. 5.3.3 – comparisons are made between the estimates produced by the standard approach and the Jacobian-scaled approach,

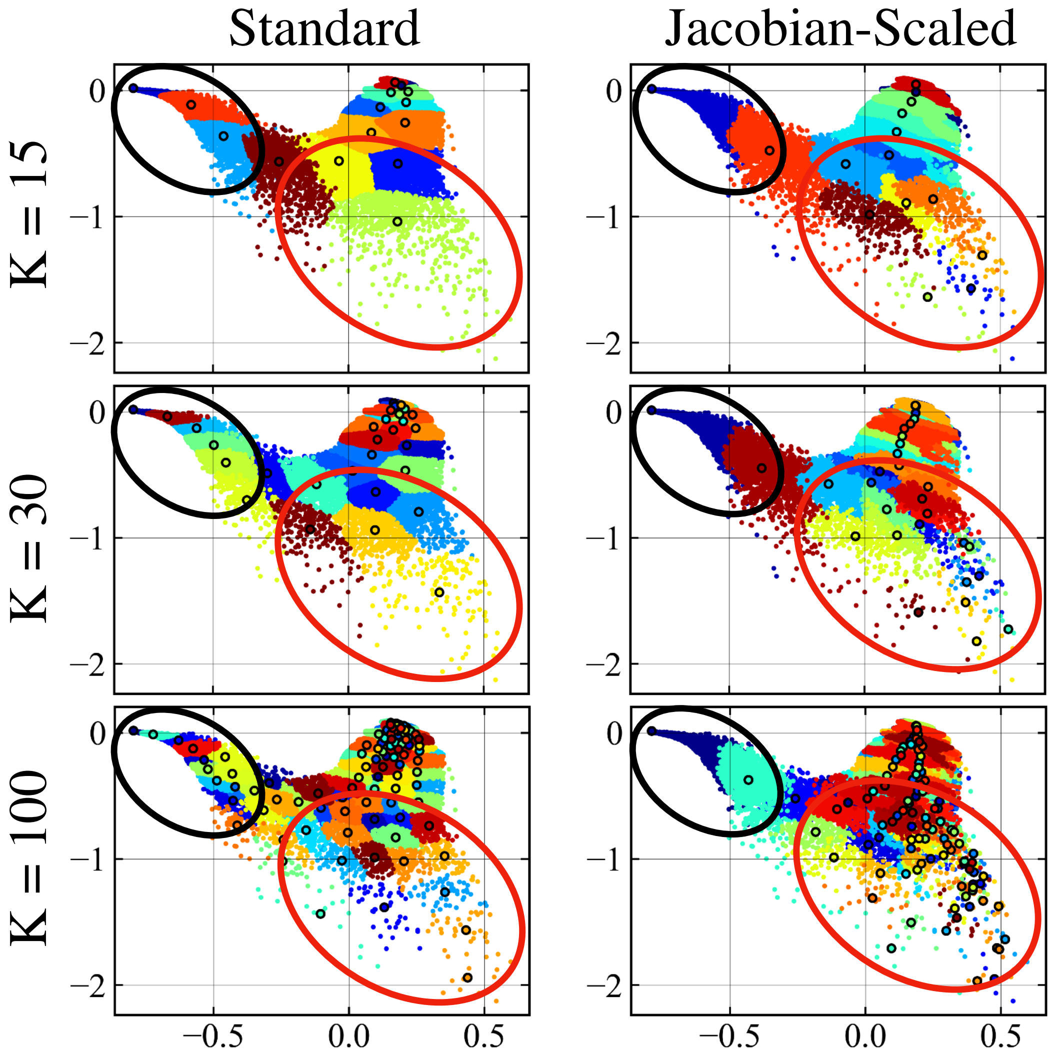


Figure 5.13: Visualizations of cluster labels from standard K-means (left column) and JS-K-means (right column) in the two-dimensional composition POD space (Eq. 5.18) for $K = 15$, 30, and 100. Centroids are denoted by larger markers outlined in black. Black circle outlines chemically non-reacting and ambient regions, whereas red circle outlines chemically reacting/sensitive regions at or near the detonation wave front.

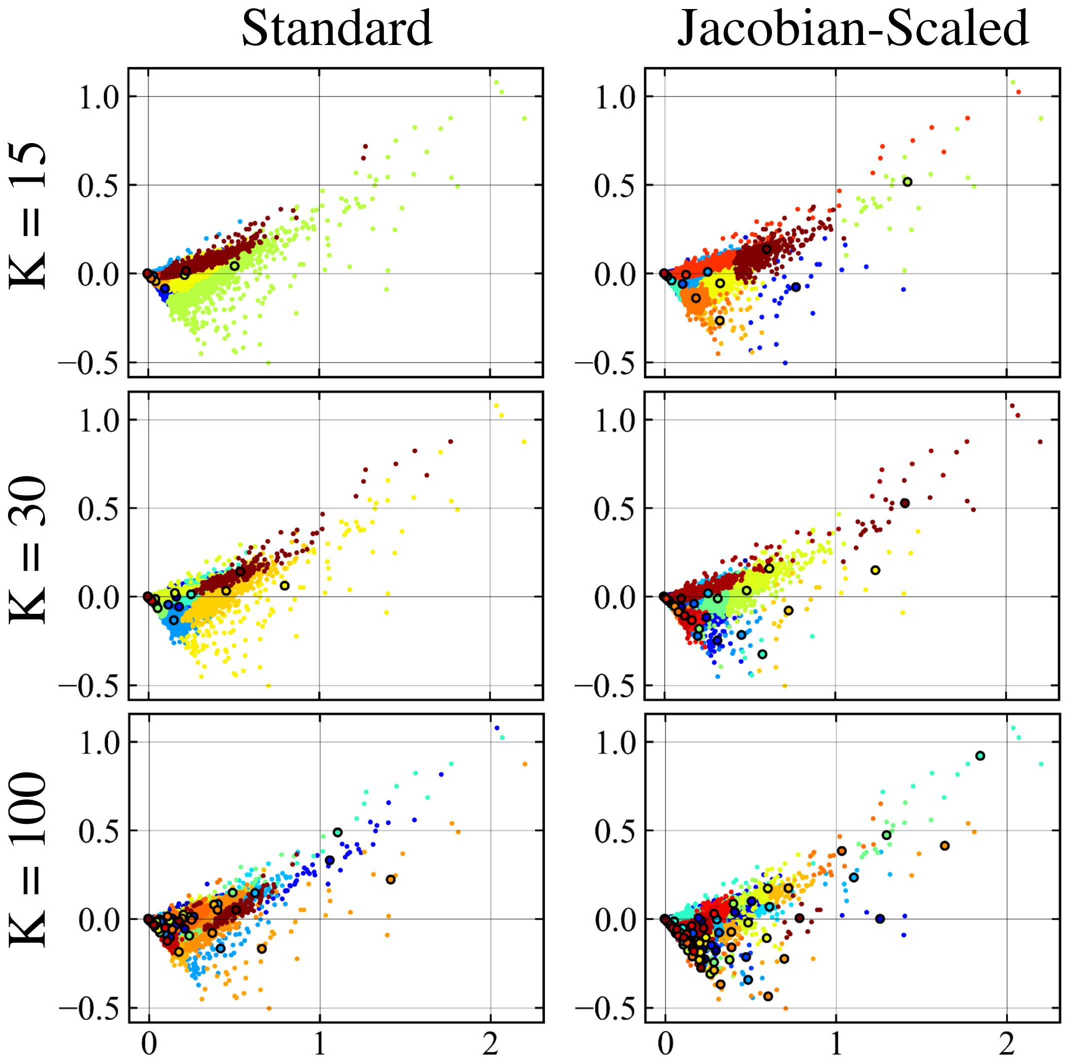


Figure 5.14: Same as Fig. 5.13, but centroids and cluster labels are plotted in the source term POD space (Eq. 5.19.

with the idea being that the source-term localization property provided by the clusters produced in the Jacobian-scaled approach should result in better predictions via the linear approximation method.

Source term predictions for temperature, ρY_{H2O}, and ρY_{H2O2} are shown in Fig. 5.15. For each component, source term predictions using the standard K-means and JS-K-means are shown using $K = 15$, 30, and 100 for assessment of cluster number impact on prediction quality. The variety of colors identify the cluster indices – points belonging to the same color utilize the same Jacobian (evaluated at the cluster centroid) to produce source term estimates via Eq. 5.15 for the corresponding samples within the cluster. Note the colors themselves are arbitrary and are provided for visual clarity purposes within the context of a single plot – that is, points of the same color appearing in different plots are not necessarily the same.

For the $K = 15$ cases, Fig. 5.15 shows general improvement in source term predictions for all three components when moving from standard K-means to the physics-guided approach. For example, the temperature source term predictions at $K = 15$ show how the error in standard K-means predictions increase for as ground-truth source term values also increase, an effect not apparent in the JS-K-means predictions. Additionally, the spread in source term values for the minor species is noticeably lower in the JS-K-means predictions. This is motivating in particular for ρY_{H2O2} – although the predictions produced from both methods here are far from ideal, the standard clustering predictions capture almost no correlation with the ground-truth data, whereas the Jacobian-scaled approach begins to extract a correlation. Figure 5.15 also shows how an increase in cluster number in general increases the quality of the source term predictions for both standard and Jacobian-scaled K-means approaches. For $K = 100$, however, the disparity in the prediction accuracy between the approaches decreases (e.g. for temperature predictions there is little noticeable difference), and for $H2O$ source term in particular, the JS-K-means fails to identify

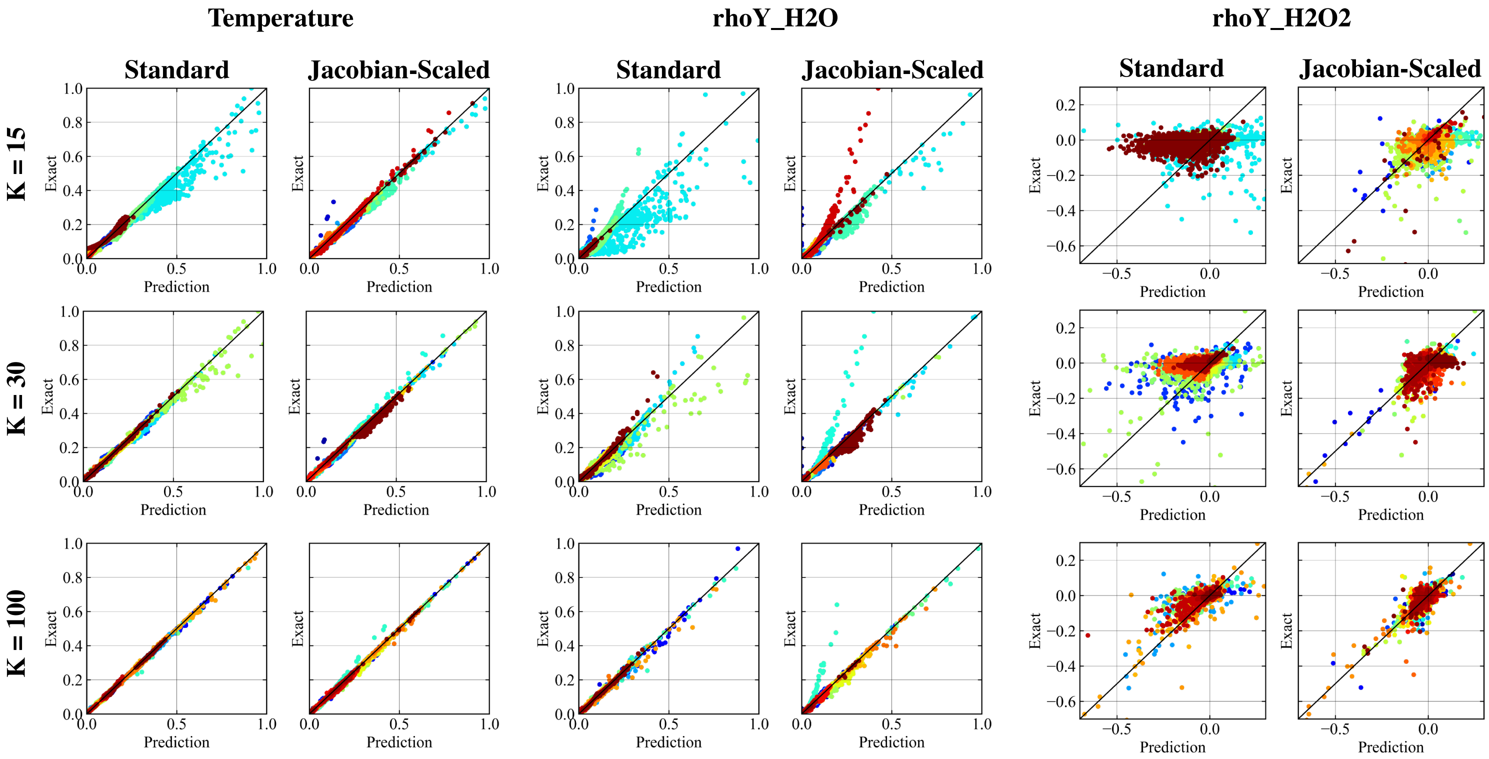


Figure 5.15: Source term predictions using standard and JS-K-means approaches for temperature source terms (left), ρY_{H2O} source terms (middle), and ρY_{H2O2} source terms (right) for $K = 15$, 30 and 100. In each plot, colors indicate cluster ownership.

one problematic cluster that is identified by the standard approach (the cyan cluster for $K = 30$ and $K = 100$) . This could be due to the fact that this cluster is either (a) attributed to a region of minimal chemical reactivity (e.g. within the black circled region in Fig. 5.13), or (b) even after the scaling procedure in Sec. 5.3.4, the JS-K-means clustering procedure is influenced disproportionately the temperature variable. These aspects will be investigated in greater detail in future work.

Ultimately, source term predictions for temperature and intermediary species mass concentrations enabled by the linear approximation in Eq. 5.12 display the advantage of the JS-K-means clustering approach from the modeling perspective, especially for lower cluster numbers. As such, the modified algorithm presented in Alg. 2 can be used a useful extension to the standard K-means approach if one has underlying knowledge of the dynamical rules in phase space via a velocity function (here the chemical source term). The physics-guided approach presented here should be interpreted as a *modifier* of the standard K-means clusters and centroids as emphasized by the burn-in procedure – given an input Voronoi partition produced by a standard K-means algorithm, the JS-K-means algorithm successful biases, or redistributes, the existing clusters towards regions of dynamical similarity by utilizing physics-based scaling matrices.

There are, however, a few weaknesses which must be mentioned: (1) the method relies on the selection of the number of clusters K as input, and (2) the improvement in source term predictions provided by the approach over standard K-means for higher cluster numbers (e.g. on the order of $K = 100$) appears to diminish. To address both of these issues, the next section introduces a hierarchical refinement strategy that eliminates the need for cluster number selection and provides an alternative pathway for embedding physics into the clustering procedure.

5.4 Hierarchical K-means Strategy

Before describing the algorithmic details, it is important to first solidify the scope of the hierarchical refinement strategy with respect to the JS-K-means approach presented in Sec. 5.3. JS-K-means is intrusive in the sense that it directly modifies the distance function (and in turn, the objective function) of the standard K-means approach and establishes a new algorithm which attempts to shift existing centroid locations towards regions of dynamical similarity. In this sense, JS-K-means algorithm presented in Sec. 5.3 is a mechanism for *redistributing* a set of centroid locations produced by the standard K-means algorithm at a user-supplied input number of clusters K.

The hierarchical K-means (H-K-means) approach presented here is fundamentally different. Starting from the same input (an initial set of K_{init} centroids produced by the standard K-means algorithm), the hierarchical refinement strategy is an outer-iteration loop that sits on top of the standard K-means algorithm – in other words, it is non-intrusive and does not directly modify the standard K-means algorithm. The termination of the outer iterations, which is governed implicitly by the error tolerance criteria, then produces a final number of clusters $K > K_{init}$. As such, although it is still guided by physics through the cluster refinement criteria, the scope of the method is different from JS-K-means presented in Sec. 5.3: instead of redistributing existing centroid locations towards regions of high chemical sensitivity, the H-K-means approach adds clusters in regions of high chemical sensitivity by means of the error estimation based refinement procedure. To emphasize this point, an illustration of the differences between H-K-Means and the previously described JS-K-Means is provided in Fig. 5.16.

The main idea of the hierarchical refinement strategy is as follows. The inputs are (1) an initial Voronoi partition of composition space produced either by the standard K-means procedure or a K-means++ initialization for a prescribed number of clusters K_{init}, and (2) a target error tolerance e_{tol}. All clusters that fail to satisfy this error tolerance are flagged for refinement, where the flagging procedure (to be described further below) is facilitated by an error estimator. Then, smaller K-means routines are run in these flagged clusters independently, the result of which increases the total number of clusters from the baseline K_{init} and adds a refinement level to the now initialized hierarchy. This procedure is then repeated on the new level until either clusters in the hierarchy satisfy the error tolerance criteria, or a user-supplied maximum level criterion is met. This strategy is inspired by both adaptive mesh refinement [27] and in-situ adaptive tabulation [263].

5.4.1 Refinement Procedure

As described above, the core of the H-K-means approach is the error estimation procedure. Because the application focus is localized source term modeling, the role

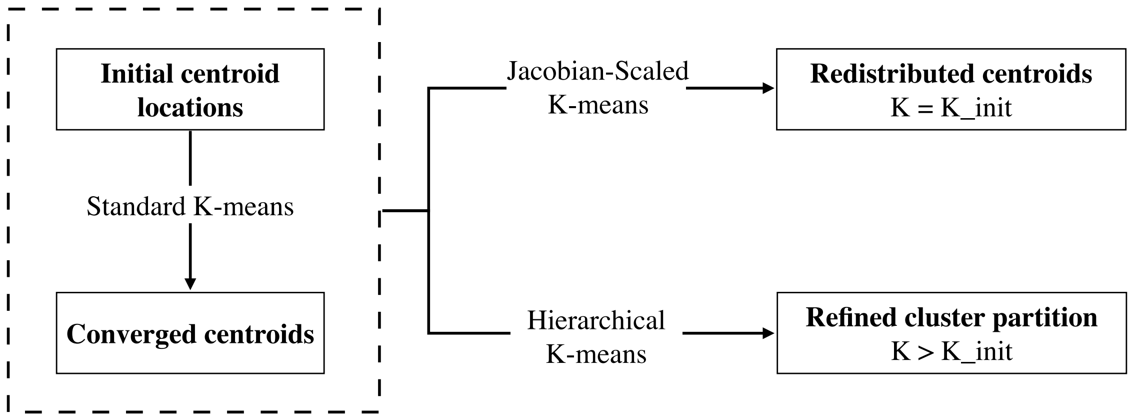


Figure 5.16: Scope of the JS-K-means approach of Sec. 5.3 and the H-K-means approach presented in this section. Both take in as input the result of a standard K-means procedure (dashed box).

of the error estimator is to predict the error incurred by the cluster-based linear approximation in Eq. 5.12. For a sample in composition space ϕ_i residing in cluster k, the estimated error is defined as

$$e_i = \|\mathbf{A}_k(\phi_i - c_k)\|_2, \tag{5.22}$$

where $\mathbf{A}_k$ is the chemical Jacobian evaluated at centroid c_k. Equation 5.22 produces the $L2$ error due to a piecewise-constant prediction for the source term (i.e. the above equation is equivalent to $\|S(\phi_i) - S(c_k)\|_2$). Note that this is the same estimation pathway used in ISAT [263]; the primary difference here is that the reference point is a cluster centroid c_k, whereas in ISAT the reference point belongs to an ellipsoid center prescribed by the singular value decomposition of the chemical Jacobian.

A reduction of the above error estimate for all samples within the k-th cluster $\mathbb{C}_k \subset \mathbb{R}^D$ then produces the cluster error estimate e_k as

$$e_k = \max(e_j), \quad j = 1, \ldots, N_k, \tag{5.23}$$

where N_k denotes the number of samples in cluster k. If $e_k \geq e_{tol}$, the cluster is selected for refinement. The refinement procedure consists of the following two steps:

1. Extract the N_k-sized subset of data from Φ residing in the k-th cluster (denoted Φ_k) using the cluster indicator matrix (Eq. 3.4).

2. Run the standard K-means algorithm (Alg. 1) on the dataset Φ_k with $K = K_{RF}$, where K_{RF} is a cluster refinement factor.

The end-result is the refinement of cluster $\mathbb{C}_k$ into K_{RF} sub-clusters, which in turn produces a hierarchy. Borrowing terminology from AMR, the procedure starts with a zeroth level, denoted L_0, populated by the output of a standard K-means procedure using K_{init} number of clusters. Performing the above refinement procedure on all K_{init} clusters instantiates the first level in the clustering hierarchy, denoted L_1. The same procedure can then be performed on the respective clusters in L_1, which then instantiates the second level L_2, and so on. This procedure is outlined in Fig. 5.17.

The hierarchy can be concisely represented by the centroid set $\mathcal{H} = \{\mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_{L_{max}}\}$, where L_{max} is the highest refinement level. The elements of $\mathcal{H}$ are themselves sets that contains the centroids in each level – for example, the quantity $\mathcal{C}_0$ contains the centroids in L_0 and is of size K_{init}. The size of $\mathcal{C}_1$ depends on the number of clusters flagged for refinement in L_0, and so on.

A so-called *composite* partition is the final output of the H-K-means algorithm. The composite partition derives a new Voronoi partition for the phase space $\mathbb{R}^D$ by applying an aggregation function over all of the $\mathcal{C}_i$ contained within the hierarchy $\mathcal{H}$. The output of this aggregation function is the the list of composite centroids, denoted $\mathcal{C}_{comp} = \{c_1, \dots, c_K\}$ – in other words, $\mathcal{C}_{comp} \leftarrow \mathcal{F}(\mathcal{H})$, where $\mathcal{F}$ is the aggregation function. The function $\mathcal{F}$ is a type of reduction operator that eliminates all centroids in $\mathcal{H}$ that have children. More specifically, each element in $\mathcal{C}_{comp}$ is a centroid that represents a cluster which contains no higher refinement levels. The centroids contained within $\mathcal{C}_{comp}$ both (a) serve as generators for a new Voronoi partition for the

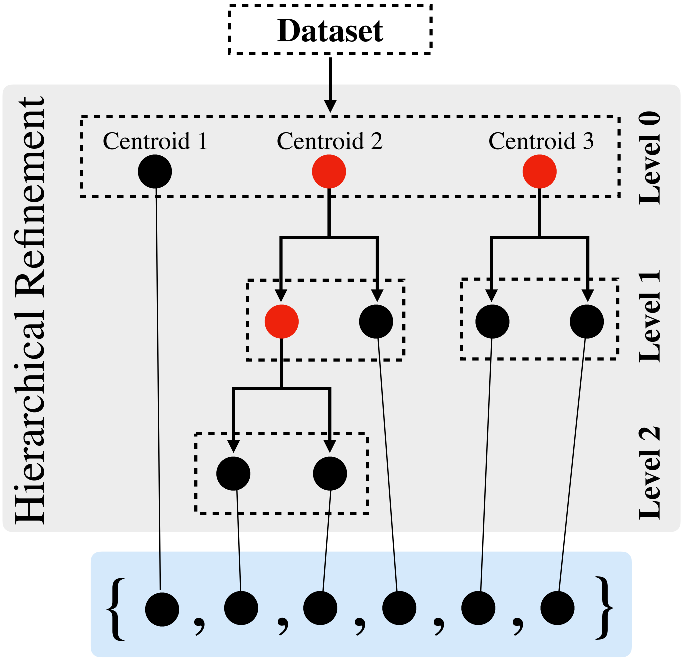


Figure 5.17: Illustration of the H-K-means approach (gray shaded region) for $K_{init} = 3$, $K_{RF} = 2$ and $L_{max} = 2$. Bold black arrows denote independent K-means calls. Red circles indicate flagged clusters/centroids that satisfy the $e_k \geq e_{tol}$ criteria, which are subsequently refined. Black circles indicate clusters/centroids that do not require refinement. Centroids grouped within the same dashed box imply a shared owner (in the case of L_0, the "owner" is simply the full input dataset). Blue shaded region contains composite set of centroids $\mathcal{C}_{comp}$ produced by aggregation function $\mathcal{F}$.

underlying D-dimensional phase space $\mathbb{R}^D$, and (b) provide the final number of clusters K produced by the hierarchical refinement strategy, where $K > K_{init}$ in the case that the initial partition on L_0 requires refinement. A visualization of the aggregation process to produce the composite centroid list is provided in Fig. 5.17. The composite centroid list $\mathcal{C}_{comp}$ is then used to facilitate the linear predictions for chemical source terms via Eq. 5.12.

5.4.2 Results

5.4.2.1 Toy Problem

To better illustrate the underlying concepts of the hierarchical K-means strategy, a simpler toy problem will be used before applying the technique to the detonation dataset. This toy problem can be considered as a 2-dimensional extension to the 1-d example case used in Sec. 5.3.5 to motivate the JS-K-means approach. Extension to two dimensions allows for a clearer visualization of the behavior of the hierarchical method.

The underlying dynamical system in this toy problem is

$$\frac{d\phi}{dt} = S(\phi) = \begin{bmatrix} \phi_1 \phi_2 \\ 0 \end{bmatrix}, \tag{5.24}$$

which produces a Jacobian matrix whose nonzero elements scale linearly with both arguments. The rate of the second phase space variable in Eq. 5.24 is set to zero to simplify the formulation as much as possible – this has the side-effect of mimicking an inert species. To populate the toy problem dataset, a each component of ϕ was sampled within the range $[0, 10]$ at 100 evenly space intervals, creating a dataset size of $N = 100^2$. All variables are scaled and nondimensinoalized using the procedure described in Sec. 5.3.4.

Figure 5.18 displays plots of the phase space variables as a function of the above nonlinear source term, illustrates the workflow of the H-K-Means approach, and provides a visualization of a simple cluster label hierarchy. As described in Fig. 5.16, a standard K-means output is fed into the hierarchical refinement procedure, which in turn refines clusters that fail to satisfy the error tolerance criterion as determined by the error estimator described in Eq. 5.22. The refinement procedure, which is a mechanism for adding additional clusters to the base phase space partition defined by the level 0 (the standard K-means output), stops until all clusters satisfy the criteria. For the example shown in Fig. 5.18, the level 0 partition consists of 10 clusters (i.e. $K_{init} = 10$) before the refinement procdeure is carried out. Upon termination of the refinement algorithm using an $e_{tol} = 0.15$, the maximum level $L_{max} = 2$ and the final cluster number for the composite set of centroids (i.e. the list of centroids that belong to clusters which have not been refined) is $K = 26$. Due to the fact that the error estimation procedure comes from a piecewise linear approximation of the source term using the centroids as tabulation points, the algorithm adds these additional clusters in regions where source term estimates are predicted to be inaccurate. The estimator implicitly assumes that points farther away from a given centroid in composition space produce linear source term predictions that are less accurate – in cases of extreme nonlinearity, this property may not be true. In practice, however, this assumption produces no major issue even for complex chemical kinetics type problems (as discussed further below), especially in cases where the target error tolerance is very small or the baseline number of clusters K_{init} is high.

The refinement phenomenon in Fig. 5.18 is similar to that observed in the JS-K-means approach (Fig. 5.4) in that the partition inherently adapts to the physics of the problem by means of invoking the dynamical system Jacobian matrix in the clustering procedure. As emphasized in Fig. 5.16, the JS-K-means approach of Sec. 5.3 accomplishes this by injecting the Jacobian matrix into the distance function for K-means, whereas the hierarchical approach described here accomplishes this through

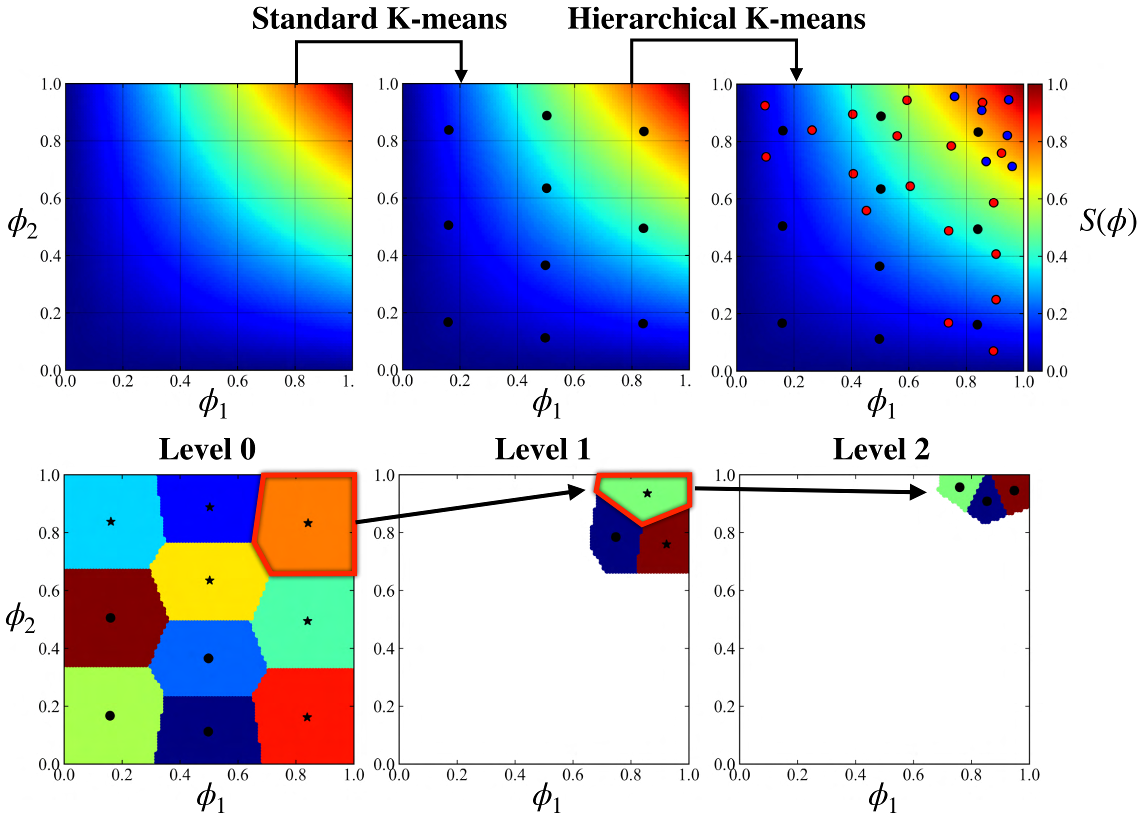


Figure 5.18: **(Top row)** Visualization of the refinement procedure output in H-K-means. Left plot shows phase space variables with source term colored, middle plot shows output of standard K-means algorithm, and right plot shows output of H-K-means with $K_{init} = 10$, $e_{tol} = 0.15$, and $K_{RF} = 3$. Black markers correspond to level 0 centroids, red to level 1, and blue to level 2. **(Bottom row)** Illustration of the hierarchy produced by H-K-means for one level 0 cluster that has been flagged for refinement (see also Fig. 5.17. Clusters flagged for refinement indicated by star markers for respective centroids.

a refinement procedure guided by a Jacobian-based error estimator.

Figure 5.19 shows the relationship between the centroid hierarchy $\mathcal{H}$ and the partition produced by the composite centroid set $\mathcal{C}_{comp}$ as a function of the error tolerance. For all error tolerances shown in Fig. 5.19, the level 0 baseline partition at $K_{init} = 10$ is the same, which allows one to assess directly the influence of the error estimator on the refinement process. The trends are consistent with expectations in that the number of output clusters increases as e_{tol} decreases. Note that the maximium refinement level is also a function of e_{tol}; for example, for $e_{tol} = 0.2$, only two total levels were produced by the refinement procedure, wheres for $e_{tol} = 0.1$, four total levels were created. Although not shown here, the number of required levels to achieve a target e_{tol} is also a function of the number of clusters K_{init} used to generate the input level 0 partition. As with methods like adaptive mesh refinement, guidelines for the prescription of the baseline level resolution are expected to be problem-dependent, which can be seen as a disadvantage. However, if the error estimation procedure is robust, the final partition due to the subsequent refinement strategy will be insensitive to the baseline level resolution for modeling purposes. Most importantly, Fig. 5.19 shows how the dropping the error tolerance indicates a noticeable allocation of clusters (or equivalently, a refinement of the composite Voronoi diagrams shown in the bottom row) in regions of high phase space sensitivity, which is the intended result.

5.4.2.2 Detonation Dataset

The H-K-means strategy is extended here to the detonation dataset described in Sec. 5.2. Note that the key trends of the hierarchical refinement procedure emphasized above for the 2-dimensional toy problem are retained in the detonation dataset. For conciseness, figures showing visualizations of refinement effects in composition space due to decreases in error tolerance are not provided. Instead, emphasis on the analysis

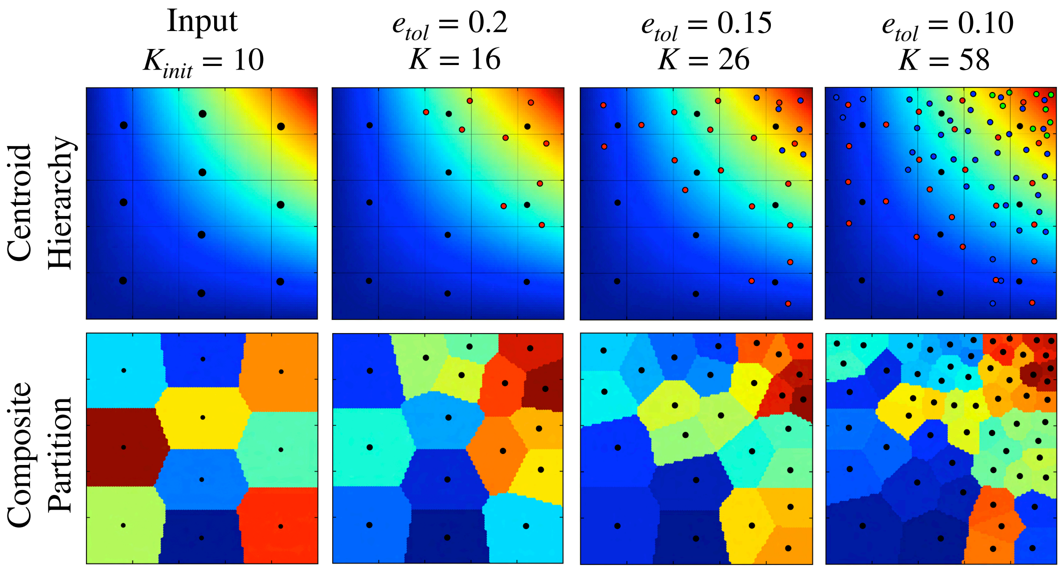


Figure 5.19: **(Top row)** Centroid hierarchy as a function of e_{tol} for the 2-d toy problem. Level 0 centroids are black, level 1 centroids red, level 2 blue, and level 3 green. **(Bottom row)** Visualization of cluster labels for composite centroid list $\mathcal{C}_{comp}$ produced by the corresponding hierarchy.

below is placed on partition comparisons between the HK-means approach and the JS-K-means approach presented in Sec. 5.3, as well as the effect of e_{tol} on source term prediction accuracy. Before proceeding, it is noted that (a) prediction results derived to the hierarchical K-means strategy come from partitions induced by the composite centroids $\mathcal{C}_{comp}$ (see Fig. 5.17 and the related discussion), and (b) all variables are scaled as per the methodology in Sec. 5.3.4 unless specified otherwise.

Figure 5.20 compares composition space partitions between the standard, Jacobian-scaled K-means, and hierarchical K-means approaches. Note that the visualizations in Fig. 5.20 utilize the POD projections described in Sec. 5.3.5. For standard and JS-K-means approaches, the partitions are shown for $K = 100$, and for the hierarchical approach, $K_{init} = 15$ was used with $e_{tol} = 0.34$ and $K_{RF} = 3$. This particular error tolerance was chosen purely to facilitate comparison with the other approaches, as it provided a final cluster number close to $K = 100$. Overall, the composite centroid distribution of the hierarchical approach for the detonation dataset behaves similarly to the JS-K-means approach of Sec. 5.3. In particular, the automated cluster refinement results in greater centroid concentrations in regions attributed to high chemical reactivity and sensitivity (red circle in Fig. 5.20).

Interestingly, the H-K-means algorithm also places centroids in regions of low temperature source term (black circles) to a much greater degree than the P-G-Kmeans algorithm – this implies that the error estimation procedure allocates importance to slightly different regions in composition space than the P-G-Kmeans approach driven by Jacobian-based distance function modification. When assessing the distribution of centroids in the source term space (bottom row of Fig. 5.20), the hierarchical procedure achieves a greater degree of centroid variation than the others. In other words, cluster refinement driven by the error estimator in Eq. 5.22 has the useful effect of maximizing centroid variation in source term space to a much greater degree than the other approaches, which is an advantage for modeling purposes. Although not shown here, dropping the error tolerance further not only increases the overall number of clusters recovered by the hierarchical approach, but further refines clusters in regions of high chemical reactivity and sensitivity.

Figure 5.21 shows source term predictions via the linear approximation in Eq. 5.12 using both the hierarchical refinement strategy and the JS-K-means algorithm of Sec. 5.3. Two sets of results are shown for the hierarchical refinement strategy: one for $K_{init} = 15$, corresponding to a coarse partition for level 0, and one for $K_{init} = 30$, which produces a finer partition for level 0. As with Fig. 5.20, to facilitate comparison with the JS-K-means output, the error tolerance was set to $e_{tol} = 0.34$ such that the output number of centroids in $\mathcal{C}_{comp}$ was close to $K = 100$.

A useful implication in Fig. 5.21 is that the value for K_{init} in the hierarchical procedure does not lead to noticeable difference in prediction accuracy across the board – this confirms that the refinement criterion is robust to the resolution of the baseline level 0 partition in composition space from which the higher levels are instantiated. Additionally, the hierarchical approach produces noticeable improvement in predic-

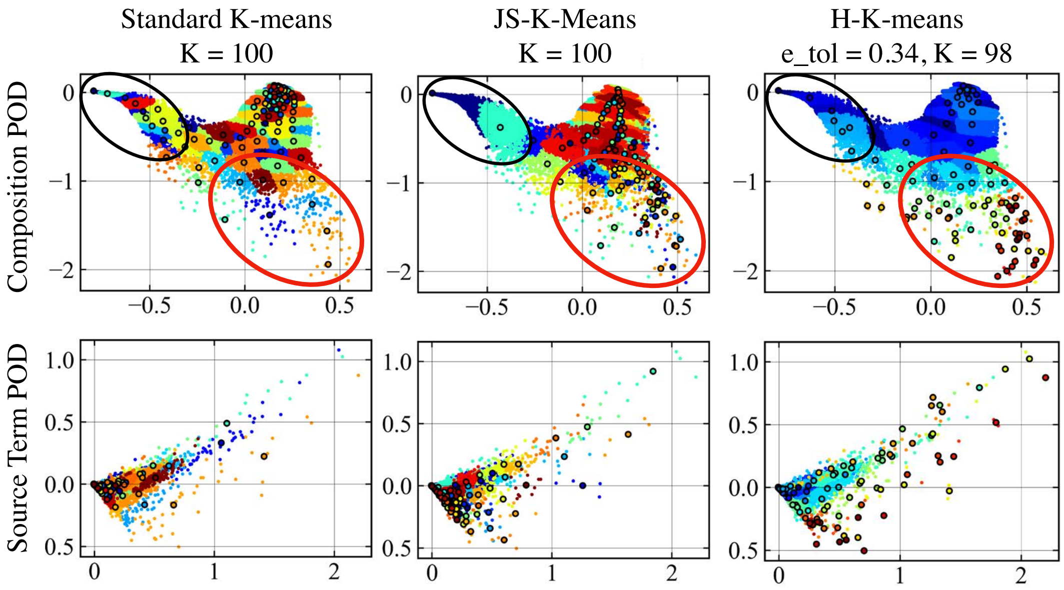


Figure 5.20: **(Left)** Standard K-means centroids for detonation dataset using $K = 100$. **(Middle)** JS-K-means centroids using $K = 100$. **(Right)** Composite H-K-means centroids using $e_{tol} = 0.34$, $K_{init} = 15$, and $K_{RF} = 3$. Centroids shown as black-outlined markers and colors indicate cluster assignments. Top row plots show projection in composition POD coordinates and bottom shows projection in source term POD coordinates (see Sec. 5.3.5 for explanation of projection method). Red circle denotes region of high chemical reactivity and black circle denotes near-ambient region.

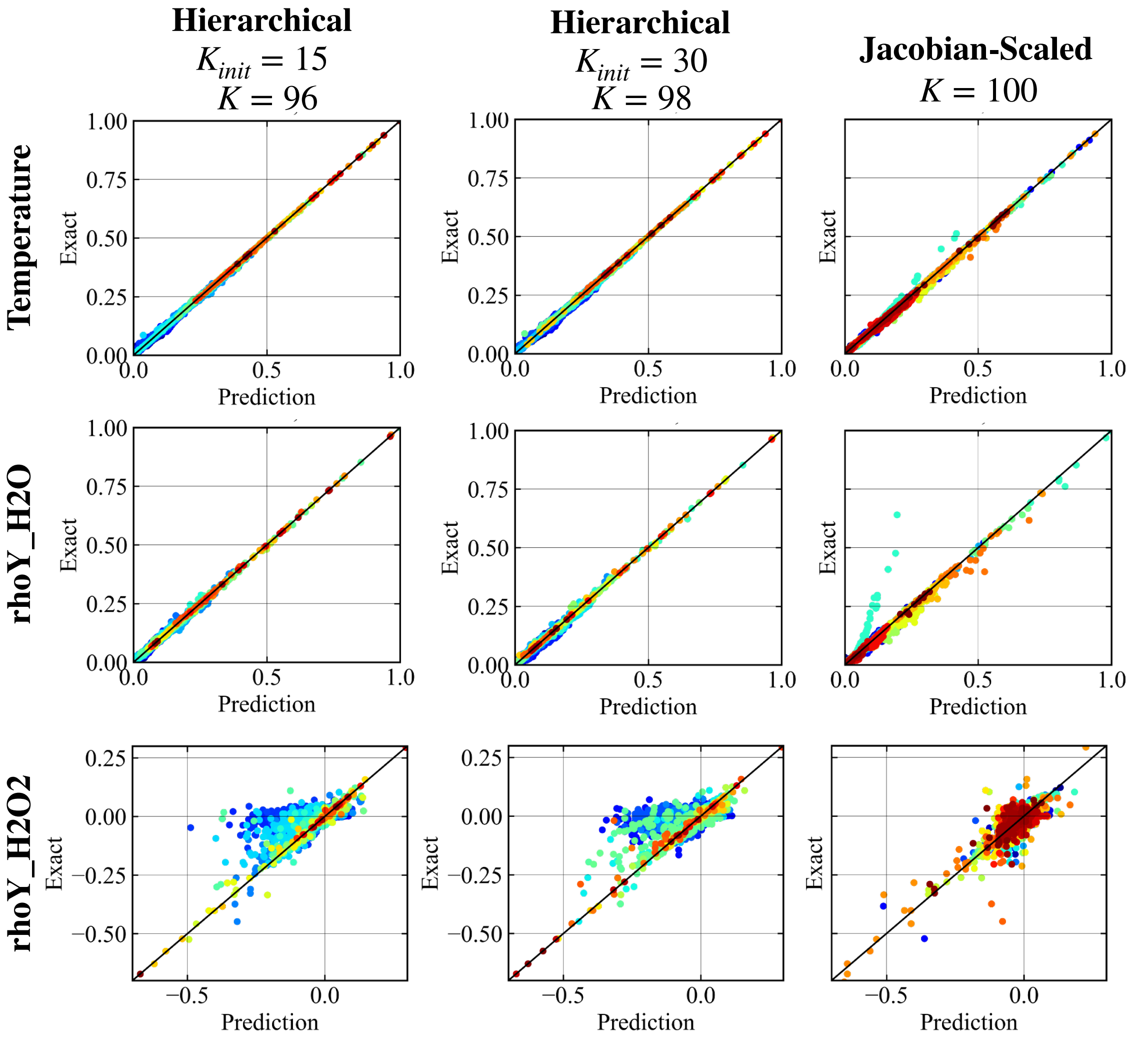


Figure 5.21: **(Left)** Source term predictions using hierarchical K-means with $K_{init} = 15$, $e_{tol} = 0.34$ and $K_{RF} = 3$. **(Middle)** Same as left, but with $K_{init} = 30$. **(Right)** Source term predictions using JS-K-means approach of Sec. 5.3 with $K = 100$. In all plots, colors denote cluster assignments and solid black line denotes perfect prediction.

tion accuracy over the JS-K-means strategy for temperature and H2O source terms (in particular, the issue with the cyan cluster in the JS-K-means approach in H2O source term prediction has been addressed by the hierarchical refinement procedure). Interestingly, at a similar value of K, the predictions in H2O2 source term appear to be slightly worse in the hierarchical approach than in the JS-K-means approach. This result, however, illuminates a potential extension of the hierarchical strategy: because the error estimation procedure as a concept is independent of the actual clustering algorithm, instead of relying on the standard K-means algorithm as the clustering tool within the refinement procedure, the JS-K-means algorithm can be used. In other words, combining the hierarchical strategy with the JS-K-means approach may lead to improved source term predictions for minor, notoriously difficult-to-predict species such as H2O2. Detailed investigation into this extension is an object of future work.

The main advantage of the hierarchical approach is the e_{tol} parameter, which eliminates the need to prescribe the number of clusters. Ideally, lowering the value for e_{tol} should produce improved source term predictions via the linear approximation in Eq. 5.12. This is indeed the case and is shown in Fig. 5.22, which plots the impact of lowering e_{tol} by one order of magnitude on the source term prediction accuracy. Prediction errors are lowered with decreasing e_{tol} across the board, especially for the problematic case of $H2O2$ source term (although there are still lingering issues at the zero-crossing). Note that although only three components are shown in Fig. 5.22, these trends apply to all remaining species concentration source terms.

To supplement the scatter plots in Fig. 5.22, root-mean squared errors (RMSE) for predicted source term for all components of ϕ are shown in Fig. 5.23 as a function of the error tolerance. The figure confirms how the error tolerance parameter serves as an effective knob for decreasing overall error at an almost linear rate for all components. Interestingly, the rate of error decrease varies for each component; a possible cause for this is the scaling procedure (i.e. the data normalization step described in Sec. 5.3.4)

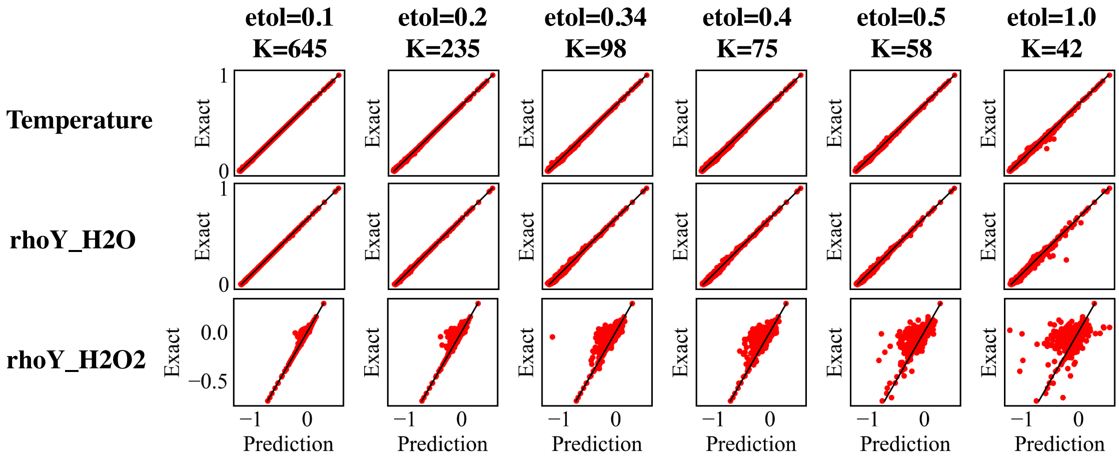


Figure 5.22: Source term predictions for temperature (top row), ρY_{H2O} (middle row), and ρY_{H2O2} (bottom row) using the hierarchical K-means strategy for various e_{tol} values. Output number of clusters K for each e_{tol} is provided in header. Solid black line denotes perfect predcition.

and the definition of the refinement criterion. To this end, a potential direction for future work is testing different methods for error estimation (e.g. applying different norms of the Jacobian-scaled distance to flag clusters for refinement).

5.5 Conclusion

In this chapter, two methods for embedding physics within the K-means clustering procedure were presented: the Jacobian-scaled K-means and hierarchical K-means. The goal of these methods is to provide alternatives to the standard K-means clustering strategy, which is based purely on composition space Euclidean distances, to account for the underlying governing equations that describe the dynamics in composition space (i.e. the chemical source terms).

The JS-K-means algorithm was formulated to minimize a modified version of the standard K-means objective; this modified objective measures within-cluster variation in chemical source terms using scaled distances in composition space. The distance scaling is accomplished by leveraging cluster-dependent chemical Jacobian matrices evaluated at centroids. Convergence of the modified algorithm based on an initial

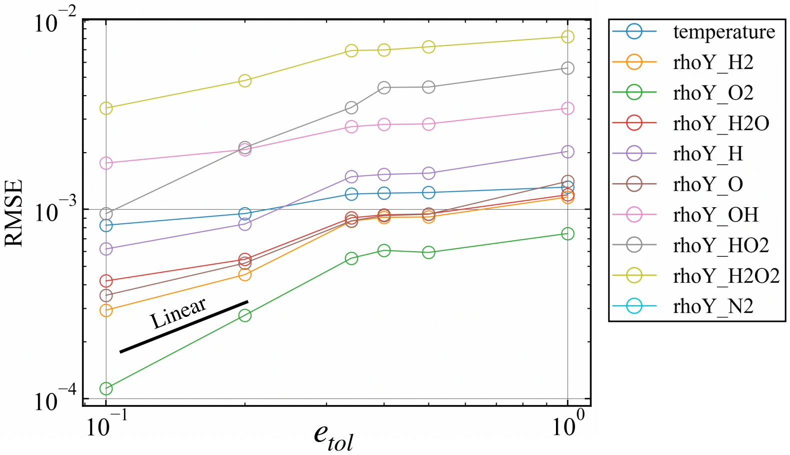


Figure 5.23: Root-mean squared errors in scaled source term predictions produced by the hierarchical K-means algorithm as a function of e_{tol} for all $D = 10$ components.

condition prescribed by the output of a standard K-means procedure (i.e. the burn-in approach) results in a redistribution, or biasing, of centroids towards regions of high chemical sensitivity. Visualization of this effect in physical space was performed via the analysis of segmented fields for a canonical detonation wave – the flowfield delineations produced by the JS-K-means algorithm pushed the standard K-means clusters towards highly stiff regions of peak heat release rate near the detonation reaction zone. Further visualization of this effect in composition space via POD projections showed how the Jacobian scaling separates (or maximizes variation between) clusters in the source-term space without explicitly using source term data during the clustering procedure, which provides improved pathways for localized kinetics modeling. To this end, linear source term predictions derived from Taylor expansions about the respective centroids demonstrated the improved modeling capability of the JS-K-means partitions.

The hiearchical K-means strategy is presented as another physics-guided clustering pathway that eliminates reliance on providing the number of clusters K as an input. Instead of modifying the distance function directly as in JS-K-means, the hierarchical strategy takes a fundamentally different approach by incorporating underlying physical knowledge into an error estimation procedure that drives cluster refinement. In other words, the approach refines select clusters in an input coarse partition produced by a standard K-means algorithm until a target error tolerance is met. An error estimation procedure based on piecewise constant predictions of chemical source term is used to flag clusters for refinement, where the refinement of a cluster is performed by running a smaller standard K-means algorithm on the subset of data belonging to the flagged cluster. The end result is a cluster hierarchy that can be concisely visualized and used for modeling purposes by extracting a so-called composite list of centroids from the hierarchy. Although the clustering mechanism is different, the cluster refinement procedure results in partitions that resemble those produced by JS-K-means in that the final clusters are inherently biased towards regions of dynamical similarity and chemical sensitivity. Usefully, the hierarchical K-means strategy allows direct tuning of the source term prediction accuracy through a reduction in the error tolerance.

Ultimately, the goal of this chapter was to extend the methodology of the classification-based regression approach described in Chapter IV by simultaneously embedding physical knowledge in the standard K-means clustering procedure and eliminating the need for ANN-based source term regression. The above pathways for physics-guided clustering successfully addressed this goal, and demonstration on the complex detonation-containing flows emphasizes the capabilities and potential of the physics-based flowfield classification strategies. Candidate next steps include comparison of source term predictions with methods like ISAT and PRISM, as well as extension of the algorithms to accommodate in-situ partition updates – to this end, part of the advantage of relying on K-means as the foundation in the above approaches is the rich literature on online K-means algorithms that render any desired in-situ extensions feasible. Additionally, an immediate next step is to utilize the source term modeling procedure from a pre-generated partition in an a-posteriori context within a high-fidelity compressible reacting flow solver. Upon solver integration, reports on time-to-solution gains due to the localized linear approximation procedure for source term modeling can then be made, and deterioration of source term evaluation times with increasing number of clusters (or conversely, smaller error tolerances for the hierarchical K-means approach) can be quantified. These details will be reported in future manuscripts.

CHAPTER VI

Summary, Conclusions, and Future Directions

6.1 Summary

Enabling long-time and high-fidelity numerical simulations of next-generation propulsion devices such rotating detonation engines and scramjets is necessary for not only probing the multi-physics behavior that governs their operation to extract physical knowledge, but also to enable robust design strategies for real-world deployment. Computational challenges that emerge in full-geometry compressible reacting flow simulations dominated by advection processes like shockwaves and detonations, however, produce prohibitive bottlenecks that require modeling strategies. In compressible reacting flow, the most problematic of these bottlenecks is the treatment of detailed chemical kinetics required to accurately describe the time evolution of species concentrations – the computational hurdles here emerge from the complexity of Arrhenius-based source term evaluations and treatment of stiffness (i.e. disparity in chemical timescales) in the associated chemical time integration schemes.

The goal of this dissertation is to provide an alternative modeling pathway, termed classification-based regression, for accelerating high-fidelity compressible reacting flow solvers targeted at the eliminating the elusive chemical kinetics bottleneck. The approach, which takes the form of a physics-guided data-driven strategy, partitions the thermochemical space using unsupervised clustering algorithms and deploys localized source term models in each partition. The novelty comes from (a) ensuring that the output segmentation produced by the partition is consistent with physical expectations in complex compressible reacting flow (e.g. the clusters identify meaningful regions within detonation wave structures in RDEs), and (b) embedding physical knowledge directly into the clustering objective function through the development of so-called physics-guided clustering procedures, such that the output partitions and classifications by design identify spatially coherent regions of dynamical similarity within the flowfield. Emphasis is placed on designing and selecting algorithms to be compatible with modern high-performance computing trends dominated by GPU-centric node architectures. This strategy was successfully used to accelerate chemical source term evaluations in complex detonation-containing flows.

Summaries of previous chapters are provided in the remainder of this section. Then, key conclusions from the classification-based regression approach are provided in Sec. 6.2. Lastly, future challenges and research directions are provided in Sec. 6.3.

Chapter I: This introductory chapter provides background on the application context of this dissertation, which is geared towards enabling robust design strategies for next generation propulsion devices like RDEs and scramjets using high-fidelity numerical simulations. More specifically, this chapter defines both the role played by compressible reacting flow simulations during the design process, and the computational challenges stemming from spatiotemporal resolution restrictions, detailed chemistry complexity, and complex geometry treatment that render the aforementioned simulations required for robust design procedures intractable. Particular emphasis was placed on the definition of ”high-fidelity” in the context of multi-physics reacting flow simulations for full-geometry propulsion devices dominated by advection; a high-fidelity simulation satisfies (a) spatial resolution requirements stemming from flow-chemistry interactions (i.e. resolution of the detonation wave structure, like induction zones and reaction zones), (b) temporal resolution requirements stemming from the wide ranges of turbulent, acoustic, and chemical timescales due to highly chaotic properties of the reacting flowfields, and (c) detailed chemistry requirements that enable accurate representation of the contribution of chemical reactions to the unsteady combustor dynamics. As all of these requirements are computationally prohibitive, this chapter concludes by describing the need for both hardware-oriented acceleration and model-oriented simulation acceleration pathways, with the latter being the focus of the dissertation.

Chapter II: Based on the target of model-oriented acceleration of compressible reacting flow simulations presented at the end of Chapter I, the goal of this chapter is to define the specific research contribution and novelty of this dissertation. To do so, a survey of existing models used throughout the numerical combustion and greater CFD community is presented via two overarching categories: physics-based models and data-based models. Alongside describing a subset of existing models of both categories in detail, the chapter describes general limitations and disadvantages of conventional physics-based and data-based modeling pathways regarding non-universality. Given these limitations, the research contribution is presented as a physics-guided data-driven modeling approach that utilizes a classification-based regression strategy for accelerated source term evaluation based on composition space partitions derived from K-means clustering. The objective of this research contribution is twofold: the first goal is to augment conventional data-based modeling strategies with constraints derived from the underlying governing equations, and the second is to design these physics-informed data-driven models to be compatible with in-situ, or online, parameter adaptation. Additional emphasis in the chapter is placed on modern high-performance computing trends to emphasize how chosen modeling frameworks should be compatible with the way hardware and HPCs have evolved in the 21st century.

Chapter III: Because the K-means clustering strategy is used as the backbone of the research contribution described in Chapter II, the goal of this chapter is to provide the basic methodology and properties of K-means clustering within the context of modeling reacting flows. The standard K-means algorithm (Lloyd's algorithm) is provided alongside discussions related to the modeling implications of the output phase space partition (the centroidal Voronoi tessellation). Both advantages and disadvantages of the standard K-means procedure are detailed, and properties of K-means clusters are illustrated by comparisons with (a) spectral clustering outputs on the illustrative half-moon dataset, (b) finite-volume based adaptive mesh refinement (AMR), and (c) proper orthogonal decomposition (POD). Comparison with spectral clustering illustrates how K-means trades cluster complexity for interpretability; comparison with AMR shows how the K-means output – a space-filling tessellation produced by centroids (generators) and a Euclidean distance function – can be interpreted as a data-adapted mesh of the phase space; comparison with POD illuminates the role centroids play from the lens of modal decompositions. Based on these qualities, the chapter provides two motivating applications to demonstrate the modeling capability of K-means clustering for turbulent reacting flow. In the first application, K-means is used to develop a data-based prognostic reduced-order model (known as cluster-based reduced order modeling, or CROM) for the analysis and prediction of flame transition in a model gas turbine combustor. In the second application, K-means is used to develop a novel decomposition strategy termed time-axis clustering, which departs from the CROM approach in that the clustering procedure is performed over individual time-series instead of snapshots. These applications are intended to be supplements/precursors to the classification-based regression approach demonstrated in the next two chapters.

Chapter IV: The goal of this chapter is to demonstrate the classification-based regression approach for unsupervised combustion regime identification and chemical source term modeling in unsteady detonation waves found in RDEs. The approach consists of two steps. In the first step (classification), a standard K-means clustering algorithm is used in the thermochemical composition space to delineate combustion regimes in the detonation wave structure via segmented flowfields. In the second step (regression), the segmented fields are used to drive the training of several ANNs, each of which produces source term estimations that are localized to the regimes identified in the first step. The intent is to show that (a) the clustering output recovers physically relevant delineations of combustion regimes in the detonation wave structure in an unsupervised manner, and (b) the source term estimations obtained from ANNs tailored to these different combustion regimes (i.e. local ANNs) are more accurate than the estimations obtained when *not* considering the clustering output during ANN training (i.e. global ANN).

Chapter V: This chapter extends the classification-based regression strategy introduced in Chapter IV by embedding physical knowledge within the K-means clustering procedure for flowfield classification. This is accomplished by augmenting the standard Euclidean distance that drives the flowfield partitioning process with the functional form for the chemical source terms (which is the modeling goal), creating the class of physics-guided K-means clustering methods. Two strategies within this class are developed in this chapter: Jacobian-scaled K-means (JS-K-means) and hierarchical K-means (H-K-means). In JS-K-means, the Euclidean distance function used in K-means clustering is modified by scaling the distance vector between two points in a given cluster with the Jacobian of the chemical source term evaluated at the centroid. This modification of the distance function induces a new objective function that is different from that of standard K-means; minimization of the new JS-K-means objective pushes clusters towards regions in composition space that reduce within-cluster variation in chemical source term without explicitly clustering over source terms directly. This method is therefore interpreted as a mechanism for *redistributing* (or biasing) a set of clusters produced by the standard K-means algorithm towards regions of increased dynamical similarity. On the other hand, instead of modifying the standard K-means algorithm directly as in the JS-K-means approach, the H-K-means strategy builds a hierarchy by refining the clusters produced by the standard K-means algorithm into sub-clusters. Clusters are flagged for refinement using an error estimation procedure based on piecewise-constant predictions of chemical source terms. Both of the above physics-guided clustering approaches are used to drive a localized modeling strategy that departs from the ANN approach used in Chapter IV, which is restricted by a training stage. Instead, because the new clustering approaches are inherently biased towards regions of dynamical similarity, the complex ANN-based source term estimation is traded for a much simpler linear extrapolation prediction based on a Taylor expansion about the cluster centroids. The physics-guided classification-based regression approach is demonstrated on a canonical channel detonation configuration.

6.2 Conclusions

High-fidelity simulations of next-generation propulsion devices like RDEs and scramjets require resolved numerical treatment of the compressible Navier-Stokes equations with detailed chemical kinetic descriptions to properly account for the coupling of chemical reactions with turbulence and shock waves. Treatment of detailed kinetics in these solvers constitutes a prohibitive computational bottleneck due to the wide range of timescales present in the building-block elementary chemical reactions. The resulting stiffness and nonlinearity stemming from the underlying dynamical system driven by the chemical source term, which comes from a linear combination of these elementary reaction rates, is notoriously difficult to deal with in all reacting flow solvers for three reasons: (1) accounting for flow-chemistry interactions requires solving a stiff ordinary differential equation in the thermochemical phase space for all cells in the computational domain at every simulation time step; (2) the arithmetic intensity of the right-hand-side evaluation for this ODE is very high (i.e. the evaluation of the chemical source term is expensive due to the complexities of the Arrhenius formulation); and (3) the costs due to time integration in (1) and source term evaluation in (2) scale super-linearly with increases in chemical mechanism complexity.

In Chapter IV, using direct numerical simulation datasets of canonical detonation configurations, a data-driven modeling procedure was developed with the goal of treating the above chemistry bottlenecks to provide long-time simulation capability for complex combustor geometries such as RDEs. The modeling approach, termed classification-based regression, consisted of two linked phases with the end-goal of accelerating chemical source term evaluations.

In the first phase (classification), classifications of instantaneous flowfields were obtained from a standard K-means clustering algorithm on thermochemical composition data obtained from simulations of a detonation wave through a linear injector array. Physical space projections of the cluster labels in composition space produced the so-called segmented fields, which were found to delineate regions of the detonation wave structure that are spatially coherent. Through an expert-guided analysis, the various regions (clusters) in the segmented field represented the ambient fuel-air mixture, a shock-separated region in the absence of fuel, a strong detonation region, and post-detonation deflagration regions within the reaction zone. Although the clustering procedure is itself unsupervised, the assignment of physical representations of each cluster in a post-processing stage allowed for the development of a useful coarse-grained perspective on detonation chemistry in physical space. Further, extending the cluster partitions produced by the LMDE dataset into to another, unseen dataset obtained from the stratified fuel-air mixture case showed how the classification strategy can be used to quantify similarities in the detonation wave structure for different configurations.

In the second phase (regression), ANNs were trained for each cluster (termed local ANNs) to output chemical source terms given the input composition variables used to produce the K-means partitions. A comparison of source term estimations obtained from the local ANNs (i.e. ANNs trained for each cluster in isolation) with the global ANN counterparts (i.e. a single ANN trained for the whole domain) showed general improvement provided by the domain-localized modeling in the training datasets. When predicting the source terms for unseen detonation waves in the same configuration at a future timestep (the LMDE testing snapshot), the improvements provided by the cluster-localized modeling became especially apparent and promising. When ambitiously extending the trained networks to the unseen data at a different configuration, it was found that although in many regions the localized model alleviated some of the large source term error variation seen in the global model, the source term predictions overall were much poorer. Despite this, the successful demonstration of this cluster-based localization of source term estimation on the unseen LMDE data is promising with regards to the role of domain-localized neural networks in the enabling of long-time simulations for complex combustion chemistry for situations in which the operating configuration does not change. Overall, the quality of local ANN predictions revealed the advantages and limitations of data-based modeling in general: the prediction quality is high for unseen flowfields belonging to the same configuration as that used to train the predictive models, and is low for unseen flowfields belonging to a different configuration (the stratified detonation case). This is consistent with expectations for purely data-driven approaches.

The classification strategy in Chapter IV utilizes the standard K-means algorithm in thermochemical composition space to produce the required flowfield delineations (segmented fields) for localized model deployment. Although the segmented fields produced by visualizing the K-means cluster assignments in physical space can be determined by expert-guided assessment to be correlated to key features within the detonation wave structure, there is an inherent issue in the classification approach of Chapter IV from the modeling perspective: by considering Euclidean distances between points in composition space to drive the partitioning procedure, there is an implicit assumption that the distance-based similarity of composition samples within the same cluster equate to similarity in the source terms of the same samples. If the underlying dynamics and sampling procedure used to populate the composition space data is nonlinear, this quality is not generally true for all relevant regions in composition space. In other words, there is no reason to believe that a partition optimized to reduce within-cluster variation in composition space is also optimized to reduce within-cluster variation in chemical source terms, especially if the source term function is highly nonlinear (which is always the case).

Two physics-guided clustering strategies developed in Chapter V – Jacobian-scaled K-means (JS-K-means) and hierarchical K-means (H-K-means) – address this issue by providing alternatives to the standard K-means clustering strategy. The new methods account for the underlying governing equations that describe the dynamics in composition space (i.e. the chemical source terms) during the clustering procedure. Ultimately, these physics-guided clustering algorithms present two separate pathways that arrive at the similar end-result, which is ensuring that the resulting clusters/partitions are intrinsically compatible with the modeling goal (source term estimation) – this in turn allows for simpler methods for source term prediction based on linearization (i.e. Taylor expansions) about the centroids, thereby eliminating the need to train ANNs for the source term estimation task.

The JS-K-means algorithm minimizes a modified version of the standard K-means objective; this modified objective measures within-cluster variation in chemical source terms using scaled distances in composition space. The distance scaling is accomplished by using cluster-dependent chemical Jacobian matrices evaluated at centroids to linearly transform the standard centroid-sample distance vector. Convergence of the JS-K-means algorithm based on an initial condition prescribed by the output of a standard K-means procedure (i.e. the burn-in approach) results in a redistribution, or biasing, of centroids towards regions of high chemical sensitivity. Visualization of this effect in physical space was performed via the analysis of segmented fields for a canonical detonation wave – the flowfield delineations produced by the JS-K-means algorithm pushed the standard K-means clusters towards highly stiff regions of peak heat release rate near the detonation reaction zone. Further visualization of this effect in composition space via POD projections showed how the Jacobian scaling separates (or maximizes variation between) clusters in the source-term space without explicitly using source term data during the clustering procedure, which in turn provides improved pathways for localized kinetics modeling. To this end, linear source term predictions derived from Taylor expansions about the respective centroids demonstrated the improved modeling capability of the JS-K-means partitions.

The hiearchical K-means strategy provides a way to eliminate the number of clusters K as an input. The approach refines select clusters in an input coarse partition produced by a standard K-means algorithm until a target error tolerance is met. An error estimation procedure based on piecewise constant predictions of chemical source term is used to flag clusters for refinement, where the refinement of a cluster is performed by running a smaller standard K-means algorithm on the subset of data belonging to the flagged cluster. The end result is the production of a cluster hierarchy, which can be concisely visualized and used for modeling purposes by extracting a so-called composite list of centroids with the help of an aggregation function. Although the clustering mechanism is different, the cluster refinement procedure results in partitions that resemble those produced by JS-K-means in that the final clusters are inherently biased towards regions of dynamical similarity and chemical sensitivity. Usefully, the hierarchical K-means strategy allows direct tuning of the source term prediction accuracy through a reduction in the error tolerance.

Overall, the classification-based regression approach for source term estimation presented in this work provides promising avenues for physics-guided data-driven acceleration of compressible reacting flow simulations. Although clustering algorithms based on Euclidean distances in thermochemical composition space (i.e. the distance function used in standard K-means) can be correlated to physical regions of interest in instantaneous reacting flowfields through expert-guided input, the concept of embedding physics into the clustering procedure by means of Jacobian-based scaling of distance vectors is a simple yet powerful tool that allows for the delineation of dynamical similarity complex flowfields without user intervention. Although the modeling strategies presented in this work were demonstrated to good success on complex detonation-containing flows, several challenges and avenues for future work remain. These are described in the next section.

6.3 Future Directions, Challenges, and Outlook

6.3.1 Next Steps for Classification-Based Regression Methodology

This dissertation presented two pathways for classification-based regression: the first (Chapter IV) equips standard K-means partitions with local ANNs for source term estimation (purely data-driven); the second (Chapter V) equips physics-guided K-means partitions with linear models for source term estimation based on Taylor expansions about the centroids. Although these strategies were demonstrated on complex detonation containing flows, demonstrating these methods on datasets sourced from full-geometry simulations of RDEs and scramjets in an a-priori setting is a target next step. A bigger challenge, and the ideal application of these methods, lies in online integration within a GPU-based reacting flow solver – discussion on this aspect is delayed to Sec. 6.3.2.

Additional next steps involve conducting more rigorous analysis of the physics-guided clustering approaches (Jacobian-scaled K-means and the hierarchical refinement strategy). This includes a more thorough analysis of the convergence properties of the modified K-means algorithm used in the Jacobian-scaled K-means clustering approach, such as investigation of centroid convergence trends with respect to number of clusters and composition dimensionality. Further, the error estimation procedure in the hierarchical refinement strategy is quite flexible and can be used in many different contexts: a promising future direction is considering an expert-selected subset of components of the feature space (a few key species) in the cluster refinement criterion instead of the entire set of features – this may be helpful for (a) extending these methods to mechanisms of very high dimensionality (on the order of 100 or 1000 species), and (b) addressing issues related to poor source term predictions for intermediary species source terms.

This dissertation focused on demonstrating a modeling approach designed to accelerate full-geometry reacting flow simulations. As such, a required next step is the direct analysis of time-to-solution gains using the classification-based regression approaches in Chapter IV and Chapter V. Note that both approaches were developed with hardware considerations in mind: the ANN forward passes used in Chapter IV and the simpler linear prediction framework of Chapter V, which consists of matrix-vector multiplications of size equivalent to the number of species, are both ideal for GPU execution. Properly assessing speedup for an apples-to-apples comparison, however, requires the development of baseline chemical kinetics libraries that execute analytic source term evaluations in a GPU-optimal fashion. Development of such a library is described in Appendix B, which provides a methodology for GPU-optimal matrix-based kinetics. On a related note, Appendix C outlines pathways for GPU treatment of chemical time integration algorithms using vectorized instantaneous source term evaluations produced either by the classification-based regression models or the analytic baseline.

The linear prediction framework enabled by the physics-guided approaches is expected to be significantly faster than the analytic baselines (the Arrhenius-based function evaluation is traded for a simple matrix-vector multiplication). Assessment of speedup in the context of ANNs, however, is nontrivial due to the fact that the ANN architecture must be specified a-priori – this was a primary motivator for adopting the physics-guided clustering approaches. As such, for a given detailed mechanism, an upper-bound on the complexity of the ANN architecture derived from the FLOPs and arithmetic intensity of the analytic baseline in Ref. [16] must be prescribed. Alongside source term evaluation times, assessment of overall time-to-solution in the classification-based regression approach must also account for the cluster assignment/labeling cost for a batch of query points, where the batch size is expected to be equivalent to the number of cells in the computational domain. Cluster assignment amounts to executing a nearest neighbor search in the composition space, the optimization of which constitutes an active area of research in its own right. In the end, for the strategy to be viable, the costs incurred by online classification cannot outweigh speedup provided by the source term estimation model. Lastly, accuracy and evaluation time comparisons with other tabulation approaches designed for in-situ kinetics modeling (i.e. ISAT and PRISM) are necessary to verify the robustness of the models presented in this dissertation.

6.3.2 Future Challenges and Recommendations

6.3.2.1 Integration with Flow Solvers

Integration of the physics-guided clustering approach into a GPU-based flow solver for in-situ kinetics modeling is a natural next step, but constitutes a significant developmental challenge. A schematic of the target online framework is shown in Fig. 2.17. Ideally, the classification-based regression framework should operate as an independent module that takes as input from the solver an instantaneous flowfield, and provides as output to the solver the chemical source terms at each grid point for time advancement. Within the module, the K-means partition should be adapted to the flowfield as it evolves, in accordance with the error tolerance and error estimation criteria provided in the H-K-means description in Chapter V. It is emphasized that true online integration ensures flowfields provided to the modeling routine are never written to disk, and are instead retained on device (i.e. GPU) memory – this is a crucial requirement for high-fidelity simulations that produce large state vectors.

A major research challenge will likely come from reducing costs in the online classification stage, which consists of batched nearest-neighbor evaluations (i.e. assigning points in composition space to their nearest centroids). Pathways to accelerate these evaluations, such as K-D trees, utilize bandwidth-limited tree traversal strategies that are known to be incompatible with GPU programming environments; conversely, brute-force search algorithms favored in GPU environments scale poorly with high cluster numbers. Lastly, an additional challenge in solver integration comes from ensuring compatibility of the physics-guided clustering algorithm (i.e. distance evaluations, centroid updates, etc.) with the domain-decomposition based representation of the input flowfield (see the discussion related to the MPI+X computing paradigm in Sec. 2.3).

6.3.2.2 Integration with Other Models

The modeling application in this dissertation focused on accelerating chemical source term evaluations via cluster-based partitions; computational limitations due to spatial resolution restrictions and chemical stiffness were unaddressed. The paragraphs below describe how the physics-guided cluster partitions can be used to address these challenges by means of combination with other modeling strategies.

Adaptive Mesh Refinement: Regarding spatial resolution restrictions, an innate quality of the physics-guided clustering strategy is its ability to track features in physical space as they evolve in time via the instantaneous segmented field. As such, the method can readily be used alongside flow solvers built around adaptive mesh refinement. This is especially useful in detonation-containing flows where regions that benefit from mesh refinement may not get picked up by conventional gradient-based cell tagging routines (i.e. the induction zone in the detonation wave structure).

Computational Singular Perturbation: Chemical stiffness (i.e. the disparity in timescales stemming from the nonlinearity of the chemical source term) and associated stiff time integration is a lingering challenge in solvers that utilize detailed kinetics that, if addressed, can result in massive overall computational gain. A physics-based ROM framework that treats this issue is computational singular perturbation (CSP, discussed in Sec. 2.4.5), which uses a basis projection to (a) decouple the fast and slow chemical timescales, and (b) eliminate the fast timescales during time integration using a manifold assumption. However, in practice, the cost of producing the time-evolving basis functions (the eigenvectors of the chemical Jacobian matrix) for each computational cell in the domain can negate the advantage provided by timescale separation. To this end, the cluster partitions can be used to eliminate the basis function evaluation cost via an extended tabulation procedure: for example, points in physical space belonging to the same cluster in composition space can share CSP basis functions. If successful, this strategy can reduce evaluation times by a factor proportional to N_C/K, where N_C is the number of computational cells in the domain and K is the number of clusters.

Autoencoders: An additional challenge touched on earlier in this section lies in scaling the clustering procedure to very complex detailed mechanisms that contain on the order of $N_S = 1000$ species or higher. Due to the curse of dimensionality, clustering in very high dimensions is challenging for several reasons, including (1) increased dependence on initialization (initial centroid locations in K-means type algorithms), (2) decreased expressive power of Euclidean-based distances, (3) higher chance of encountering complex/irregular cluster structures, and (4) long nearest neighbor search times. Mechanism reduction strategies are ultimately required to address these issues. Although separate modeling approaches designed to reduce mechanism size have been widely explored in the numerical combustion community from both physics-based and data-based perspectives, the order reduction provided by these methods does not guarantee a cluster-compatible latent space (or reduced composition space). To this end, methods that integrate dimensionality reduction objectives with clustering objectives into a unified framework can be used to scale the classification-based regression strategy to high N_S. An example of this is the class of deep clustering methods which utilize autoencoders for dimension reduction (see Sec. 2.5.5).

Time-Axis Clustering: Time-axis clustering was presented in Sec. 3.5 as a novel modal decomposition strategy that produces a segmented field similar to the clustering procedures used in Chapters IV and V. The fundamental difference is that it executes K-means over time series data sampled from an ensemble of spatial collocation points, where the ensemble size is equal to the number of grid points used to discretize the spatial domain. As such, instead of centroids representing local averages of thermochemical composition samples (Chapters IV and V), centroids in the time-axis decomposition represent conditionally averaged time series for an observable function. Although the time-axis concept was introduced and demonstrated briefly on a combustor dataset in Sec. 3.5, further applications of the technique are warranted due to its ability to correlate these conditionally averaged time series with unique locations in physical space. The concept of deriving order reduction from conditional averages of underlying dynamics is also related to ideal large-eddy simulation formulations (see Sec. 2.4.1) – solidifying this relationship is a promising avenue for exploring the potential time-axis clustering.

6.3.3 Closing Remarks

The abundance of data in modern times has cemented the role of data-driven modeling as a pathway for accelerating high-fidelity numerical simulations of fluid flows. This is especially useful for guiding both design processes and physical understanding of next-generation propulsion concepts like rotating detonation engines and scramjets, which are known to exhibit highly complex unsteady behavior stemming from the multiscale interactions between turbulence, shockwaves, and chemical reactions. By embedding underlying physical knowledge into the data-driven modeling workflow and ensuring algorithms can deployed in online (or in-situ) settings within flow solvers, data-driven models have the powerful ability to treat issues of non-universality present in both purely physics-based and purely data-based modeling alternatives.

It is important to recognize that the rapid rise in the availability of data in the 21st century is also reflected in the contemporaneous evolution of hardware via the widespread adoption of energy-efficient GPUs. As such, when designing new data-driven modeling approaches, the numerical combustion community must ensure that developed algorithms are compatible with evolving trends in hardware such that modeling approaches can be realistically deployed for solver acceleration purposes, and are compatible with the now GPU-dominated high performance computing resources.

Ultimately, research into data-driven modeling for compressible reacting flow applications is still in its infancy – the techniques described in this dissertation provide one pathway by which physics-guided data-driven modeling can be used to push new, more efficient hypersonic energy generation devices into widespread usage.

APPENDICES

APPENDIX A

Derivation of Centroid Update Rule

The sections below outline derivations of the centroid update rule for three versions of the K-means objective:

1. The standard K-means objective that utilizes the vanilla Euclidean distance measure (Sec. A.1).
2. A modified K-means objective that scales sample-centroid distances by a fixed scalar value that is independent of the centroid locations (Sec. A.2).
3. A modified K-means objective that scales sample-centroid distances by a centroid-dependent scaling variable (Sec. A.3); this is the same modification used in the Jacobian-scaled K-means approach in Sec. 5.3.

As will be seen below, the derivations show that the centroid update rule obtained from minimizing (1) and (2) comes from the average of within-cluster samples – i.e. scaling the K-means objective by a constant factor does not change the standard centroid update rule as expected. However, for approach (3), which is the same modification used in the Jacobian-Scaled K-means formulation in Sec. 5.3, minimization of the modified objective produces a centroid update rule that deviates from the within-cluster average by a residual proportional to the within-cluster variance of samples.

For illustrative purposes and to simplify the derivations, the input samples here are scalars (i.e. $D = 1$); derivation trends and main takeaways from the scalar case are expected to apply in general to higher dimensional cases. Also, to facilitate clearer and more readable derivations, the notation used below deviates from that used in Chapter III, and instead follows the notation used in Ref. [37].

The input data samples are denoted $\phi_i \in \mathbb{R}$, $i = 1, \ldots, N$, where N is the number of samples. The centroids are denoted $c_k \in \mathbb{R}$, $k = 1, \ldots, K$, where K is the number of clusters. The set of centroids is given by $c = \{c_1, c_2, \ldots, c_K\}$.

The objective in the K-Means algorithm is to find a c that minimizes the within-cluster variance. This can be expressed in the obective function $E(c)$, where

$$E(c) = \sum_i \frac{1}{2} \left\| \phi_i - c_{s_i(c)} \right\|_2^2 = \sum_i \frac{1}{2} \left(\phi_i - c_{s_i(c)} \right)^2. \tag{A.1}$$

In Eq. A.1, $s_i(c)$ encodes the centroid index that is closest to the sample ϕ_i. In other words, $s_i(c) = k$ if sample ϕ_i is closest to centroid c_k in the Euclidean sense. Equation A.1 is equivalent to the standard K-means objective provided in Eq. 3.1 in the scalar case – the factor of 1/2 is inconsequential and is included above for convenience. Additionally, the function of $s_i(c)$ is equivalent to the function of the assignment matrix $\mathbf{L}$ in Eqs. 3.1 and 3.4.

In the standard K-means approach, the goal is to produce a set of K centroids that minimizes the above objective. To accomplish this, one path is to recover an centroid update equation from gradient descent as

$$\Delta c = \epsilon \frac{\partial E(c)}{\partial c}, \tag{A.2}$$

where ϵ is a so-called learning rate. Although we can proceed in the gradient descent context, a more intuitive formulation comes from the expectation-maximization (EM) perspective [37], which boils down to the following question: **given some previous**

value of the centroids, what are the values of $s_i(w)$ that minimize the objective? The derivations below proceed in the context of this question.

A.1 Standard K-means

We can cast the above question in the following cost function, which is related to the original objective defined in Eq. A.1:

$$Q(c, c') = \sum_i \frac{1}{2} \left\| \phi_i - c'_{s_i(c)} \right\|_2^2 = \sum_i \frac{1}{2} (\phi_i - c'_{s_i(c)})^2. \tag{A.3}$$

In Eq. A.3, the current set of centroids is c and the next (new) set of centroids is c'. The goal is to find an update rule for c'_k – the k-th centroid at the next iteration – that minimizes the objective/cost function in Eq. A.3. This is referred to as "standard" K-means (see Chapter III) because the objective function in Eq. A.3 utilizes the usual Euclidean distance.

The analytic solution to the minimization comes from solving the following algebraic equation:

$$\frac{\partial Q(c, c')}{\partial c'_k} = 0. \tag{A.4}$$

Because K-means is a hard clustering method, there are two conditions: $s_i(c) = k$ and $s_i(c) \neq k$. If $s_i(c) = k$, the partial derivative in Eq. A.4 becomes

$$\begin{aligned} \frac{\partial Q(c, c')}{\partial c'_k} &= \frac{\partial}{\partial c'_k} \left[\sum_i \frac{1}{2} (\phi_i - c'_k)^2 \right] \\ &= \sum_i \frac{\partial}{\partial c'_k} \left[\frac{1}{2} (\phi_i - c'_k)^2 \right] \\ &= \sum_i c'_k - \phi_i. \end{aligned} \tag{A.5}$$

If $s_i(w) \neq k$, the partial derivative in Eq. A.4 is zero. Combining these conditions leads to the expression

$$\frac{\partial Q(c,c')}{\partial c'_k} = \sum_i \begin{cases} c'_k - \phi_i & \text{if } s_i(c) = k, \\ 0 & \text{otherwise.} \end{cases} \rightarrow \frac{\partial Q(c,c')}{\partial c'_k} = \sum_{i:s_i(c)=k} (c'_k - \phi_i). \tag{A.6}$$

Solving for Eq A.4 results in

$$\begin{aligned} \frac{\partial Q(c,c')}{\partial c'_k} &= \sum_{i:s_i(c)=k} (c'_k - \phi_i) = 0, \\ \sum_{i:s_i(c)=k} c'_k &= \sum_{i:s_i(c)=k} \phi_i, \end{aligned} \tag{A.7}$$

which ultimately provides the familiar centroid update rule:

$$c'_k = \frac{1}{N_k} \sum_{i:s_i(c)=k} \phi_i, \text{ where } N_k = \sum_{i:s_i(c)=k} 1. \tag{A.8}$$

In other words, the centroid at the next iteration c'_k is the within-cluster sample mean using labels from the previous iteration – convergence of the iterative procedure minimizes the target objective in Eq. A.3, which was the starting point. This is the update rule used in both Alg. 1 and Alg. 2.

A.2 Constant Scaling Factor

Here, the standard K-means objective is modified slightly with a constant-valued, linear scaling factor a:

$$Q(c,c') = \sum_i \frac{1}{2} \left\| a(\phi_i - c'_{s_i(c)}) \right\|_2^2 = \sum_i \frac{1}{2} (a\phi_i - ac'_{s_i(c)})^2. \tag{A.9}$$

Constant-valued here means a is set once a-priori and fixed – since both data points ϕ_i and centroids c_k are scalars, a in this case is also a scalar. In the case of $s_i(c) = k$, the partial derivative becomes

$$\begin{aligned}\frac{\partial Q(c,c')}{\partial c'_k} &= \frac{\partial}{\partial c'_k}\left[\sum_i \frac{1}{2}(a\phi_i - ac'_k)^2\right] \\ &= \sum_i \frac{\partial}{\partial c'_k}\left[\frac{1}{2}(a\phi_i - ac'_k)^2\right] \\ &= \sum_i a^2(c'_k - \phi_i).\end{aligned} \tag{A.10}$$

Again, note that if $s_i(c) \neq k$, the partial derivative $\frac{\partial Q(c,c')}{\partial c'_k} = 0$. This leads to

$$\frac{\partial Q(c,c')}{\partial c'_k} = \sum_i \begin{cases} a^2(c'_k - \phi_i) & \text{if } s_i(c) = k, \\ 0 & \text{otherwise.}\end{cases} \rightarrow \frac{\partial Q(c,c')}{\partial c'_k} = \sum_{i:s_i(c)=k} a^2(c'_k - \phi_i). \tag{A.11}$$

Because the parameter a is nonzero, solving $\frac{\partial Q(c,c')}{\partial c'_k} = 0$ for c'_k produces the same update rule as the standard case (Eq. A.8).

A.3 Centroid-Dependent Scaling Factor

The standard K-means objective is modified here with a *centroid-dependent* scaling factor $A(c_k) \in \mathbb{R}$:

$$Q(c,c') = \sum_i \frac{1}{2}\left\|A(c'_{s_i(c)})\left(\phi_i - c'_{s_i(c)}\right)\right\|_2^2 = \sum_i \frac{1}{2}\left(A(c'_{s_i(c)})\phi_i - A(c'_{s_i(c)})c'_{s_i(c)}\right)^2. \tag{A.12}$$

The scaling factor $A_k = A(c'_k)$ is evaluated at a specific centroid location and is directly analogous to the role played by $\mathbf{A}_k$ in the Jacobian-scaled K-means formulation used in Chapter V, Sec. 5.3. This means that the scaling factor A_k dynamically adapts to the movement of the centroids during the K-Means iterations. As alluded in Sec. 5.3.2, the derivations below show that the standard centroid update incurs an error when augmenting the distance function in the objective with a centroid-dependent linear scaling, as in Eq. A.12.

In the case of $s_i(c) = k$, the partial derivative becomes

$$\begin{aligned}\frac{\partial Q(c,c')}{\partial c'_k} &= \frac{\partial}{\partial c'_k}\left[\sum_i \frac{1}{2}\left(A(c'_k)\phi_i - A(c'_k)c'_k\right)^2\right] \\ &= \sum_i \frac{\partial}{\partial c'_k}\left[\frac{1}{2}\left(A(c'_k)\phi_i - A(c'_k)c'_k\right)^2\right] \\ &= \sum_i \left(A(c'_k)\phi_i - A(c'_k)c'_k\right)\underbrace{\frac{\partial}{\partial c'_k}\left[A(c'_k)\phi_i - A(c'_k)c'_{s_i(c)}\right]}_{\Psi}\end{aligned} \tag{A.13}$$

Expanding for the term denoted Ψ in the above equation yields

$$\begin{aligned}\Psi &= \frac{\partial}{\partial c'_k}\left[A(c'_k)\phi_i - A(c'_k)c'_{s_i(c)}\right] \\ &= \frac{\partial}{\partial c'_k}\left[A(c'_k)\phi_i\right] - \frac{\partial}{\partial c'_k}\left[A(c'_k)c'_k\right] \\ &= \phi_i\frac{\partial A(c'_k)}{\partial c'_k} - \left(c'_k\frac{\partial A(c'_k)}{\partial c'_k} + A(c'_k)\right) \\ &= \frac{\partial A(c'_k)}{\partial c'_k}\left(\phi_i - c'_k\right) - A(c'_k).\end{aligned} \tag{A.14}$$

Plugging back in to Eq. A.13:

$$\begin{aligned}\frac{\partial Q(c,c')}{\partial c'_k} &= \sum_i \left(A(c'_k)\phi_i - A(c'_k)c'_k\right)\underbrace{\left(\frac{\partial A(c'_k)}{\partial c'_k}\left(\phi_i - c'_k\right) - A(c'_k)\right)}_{\Psi} \\ &= \sum_i -A(c'_k)\left(A(c'_k)\phi_i - A(c'_k)c'_k\right) + \frac{\partial A(c'_k)}{\partial c'_k}(\phi_i - c'_k)\left(A(c'_k)\phi_i - A(c'_k)c'_k\right) \\ &= \sum_i \underbrace{\frac{\partial A(c'_k)}{\partial c'_k}A(c'_k)(\phi_i - c'_k)^2}_{\text{Variance contribution}} - \underbrace{A(c'_k)^2(\phi_i - c'_k)}_{\text{Fluctuation contribution}}.\end{aligned} \tag{A.15}$$

Note that in Eq. A.15, the partial derivative contains the same fluctuation contribution as Eq. A.10, the constant scaling case. What is introduced by the additional requirement of centroid-dependency in the scaling factor, however, is an additional contribution that (a) depends on a gradient of $A(c'_k)$, and (b) scales with the within cluster variance expressed as $(\phi_i - c'_k)^2$. This modification can also be interpreted as the inclusion of a higher-order term in the form of an additional within-cluster moment.

Recall again that in the case of $s_i(c) \neq k$, we get the usual $\frac{\partial Q(c,c')}{\partial c'_k} = 0$. As such, an iterative solution to the minimization problem requires solving the following equation for c'_k:

$$\frac{\partial Q(c,c')}{\partial c'_k} = \underbrace{\sum_{i:s_i(c)=k} \frac{\partial A(c'_k)}{\partial c'_k} A(c'_k)(\phi_i - c'_k)^2}_{\text{Variance contribution}} - \underbrace{\sum_{i:s_i(c)=k} A(c'_k)^2(\phi_i - c'_k)}_{\text{Fluctuation contribution}} = 0. \tag{A.16}$$

The key takeaway is that if the variance contribution diminishes, the update rule for the centroids becomes the standard update of Eq. A.8. The above equation can therefore be interpreted as deviation from the standard centroid update rule by a residual proportional to the variance contribution. As implied by Eq. A.16, a variance contribution of zero would require one the following conditions to be met:

- The gradient of the scaling factor within the cluster is zero.
- The within-cluster variance is zero.
- The scaling factor itself is zero.

If the scale factor $A(c_k)$ is derived from the Jacobian of a chemical source term (as in the Jacobian-scaled K-means procedure described in Chapter V), the first condition cannot be valid in regions of high chemical sensitivity. Further, within this context, the second and third conditions are met only in trivial equilibrium and ambient conditions in which there is no chemical contribution to the dynamics (i.e. zero source term). As such, a centroid update that does not take into account the variance contribution in Eq. A.16 cannot guarantee monotonic convergence of the modified objective function in Eq. A.12 – this is consistent with the non-monotonic convergence trends observed for the Jacobian-scaled K-means algorithm in Sec. 5.3.

APPENDIX B

Matrix Formulations of Chemical Kinetics for Acceleration on GPUs

The objective of this Appendix is to provide a GPU-optimal analytic baseline for source term evaluation required to assess the computational benefit of the classification-based regression algorithms presented in Chapters IV and V. In a nutshell, the method presented here achieves optimal GPU evaluation of the chemical source terms by recasting the analytic equations for the reaction rates in a matrix-oriented fashion that resembles neural network layers [16]. The idea is to extract the computational benefit of batched matrix-multiplications over cells provided by frameworks like neural networks without the need for a training stage. This resulted in the development of a CUDA-based library, `UMChemGPU`, which consists of wrappers around lower level NVIDIA cuBLAS and cuSPARSE matrix multiplications to drive the chemical source term evaluations. A schematic of the library-solver interface is shown in Fig. B.1, and the end-user operating scope upon integration into a flow solver is shown in Fig. B.2. It is a separately compiled module designed to be linked to any reacting flow solver to enable GPU-optimal kinetics treatment for both instantaneous chemical source term evaluation (discussed here) and chemical time integration (discussed in Appendix C – see Refs. [16, 30, 340] for more detail on this matter, and for an analysis of the solver performance gain provided by `UMChemGPU` when treated as a drop-in replacement for

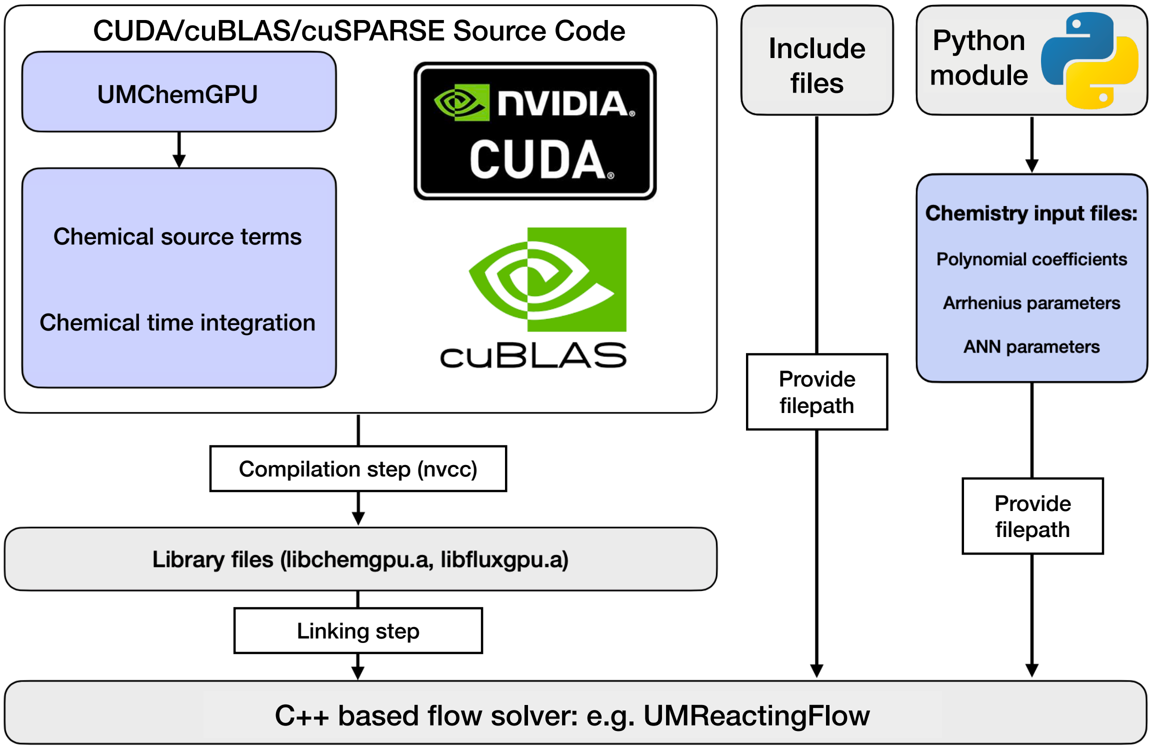
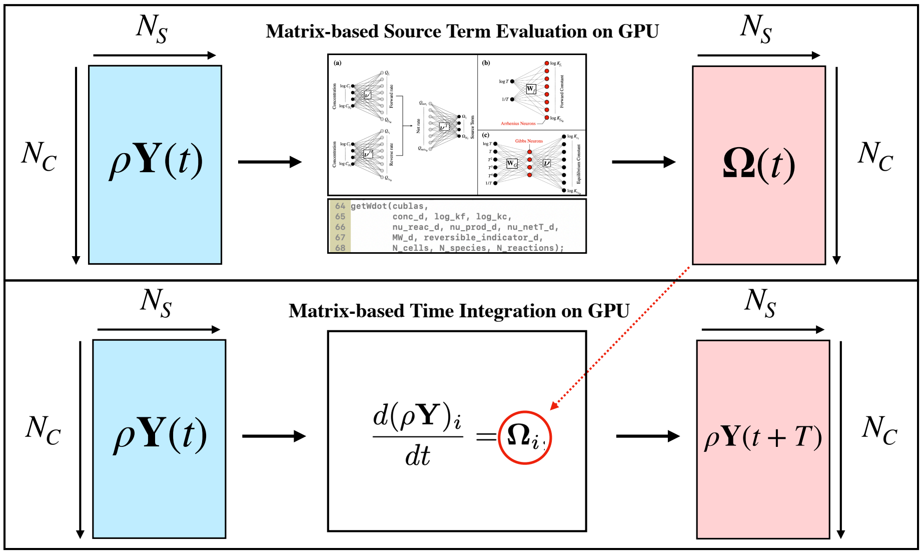


Figure B.1: UMChemGPU library interface. Wrappers around CUDA-based subroutines are compiled and then linked to the flow solver. Python module extracts key parameters from mechanism file via Cantera library [109] (e.g. stoichiometric coefficients, polynomial coefficients, etc.).

conventional CPU-based kinetics libraries.

The method presented here is important not only for providing an ideal GPU-based baseline for separate kinetics modeling approaches, but also for guiding the development of new models that seek to achieve simulation acceleration via compatibility with this GPU-optimal approach. For example, the analytic approach presented here provides an upper bound on floating point operations (FLOPs) and arithmetic intensity for source term evaluation that can guide the design of ANN architectures used in Chapter IV (the ANN forward pass should not be more expensive than the GPU-optimal baseline). Additionally, GPU-optimal source term evaluations are required for integrating the source term tabulation approach based on physics-guided clustering in Chapter V into flow solvers, as analytic source terms (as well as Jacobians) are required at the reference points (which are the cluster centroids in Chapter V).



```
getWdot(cublas,
        conc_d, log_kf, log_kc,
        nu_reac_d, nu_prod_d, nu_netT_d,
        MW_d, reversible_indicator_d,
        N_cells, N_species, N_reactions);
```

Figure B.2: End-user operating scope of UMChemGPU. **(Top)** Matrix-based instantaneous chemical source term evaluation, which is described in this Appendix. Blue matrix on left is matrix of species concentrations, and red matrix is corresponding chemical source terms (although not shown in schematic, temperature source term also provided). N_C is number of cells and N_S number of species. **(Bottom)** Matrix-based chemical time integration routine that uses matrix-based rate evaluations, which is described in detail in Appendix C.

Note that other approaches for GPU-offloading for chemical kinetics have been explored in detail in recent years [68, 221, 308], and their implementation into high-fidelity parallel solvers has also been demonstrated [249]. The novelty of the method presented here is the emphasis on ensuring vectorization over number of cells, species, and reactions via matrix formulations which are ideal for GPU computing environments. Additionally, emphasis here is placed on the importance of throughput analysis, which ensures that the method is utilizing the GPU hardware to its fullest extent. In the end, upon integration into flow solvers, the method (via the `UMChemGPU` library) displays overall time-to-solution gains that increase as (a) the number of cells offloaded to the GPU increases, and (b) the mechanism complexity increases [30, 340].

The remainder of this Appendix proceeds as follows. In Sec. B.1, the methodology for the matrix-inspired formulation is presented in the language of artificial neural networks, and a classification of reaction types that facilitates the GPU performance analysis is provided. In Sec. B.2, the GPU performance is assessed in detail from a compute time and throughput perspective, the costs of individual reaction types are assessed, and a pathway for improving the speedup for very large mechanisms is provided.

B.1 Methodology

This section first summarizes the chemical kinetic equations from a matrix-based perspective (Sec. B.1.1), and then discusses the data structure and organization of the matrices used in the GPU computations (Sec. B.1.2). The matrix-based formulations are presented in the language of traditional artificial neural networks (ANNs) where appropriate. Conveying the exact formulation of the source term computation with ANNs is intended to inform the reader that the form of kinetics equations as-is is efficiently described via neural network layers without any need for training or modeling. For this reason, the ANN connections can lead to valuable insights regarding the general interpretation and potential design of GPU-optimal chemical mechanisms. Additionally, the neural network connections open pathways for constrained modeling approaches, here called "approximate ANNs", which can be designed to reduce the overall computational effort undertaken during the exact source term computation at the cost of perfect accuracy. Although it does not fall into the main scope of this Appendix, additional details on the approximate ANN formulation are provided Ref. [16].

Of special importance in Sec. B.1.2, and tied to the matrix data structures explained therein, is the distribution of different *reaction types* present in a given mechanism. In general, different algorithms are required for different reaction types—some allow for matrix formulations and others do not. As such, a characterization of chemical mechanisms based on the distribution of these reaction types is presented to provide a pathway for assessing: (a) how beneficial the matrix representation can be for a particular mechanism; and (b) how the prevalence of specific reaction types can positively or negatively impact the GPU-derived speedup. The main reason for introducing the reaction-type classification is to bring forward the idea that, in the determination of GPU speedup, mechanism species and reaction numbers are not the only factors; the complexity of the individual reactions themselves also plays an important role.

B.1.1 Matrix-Based Kinetics Equations

The analytic equations for chemical source terms are repeated here from Sec. 2.2 for convenience, and are then recast in the matrix-based formulation. In the following, the quantities N_C, N_S, and N_R denote the batch size (which can be interpreted as the number of reacting cells in a domain offloaded to the GPU), number of species, and number of reactions, respectively. Unless otherwise indicated, matrices are denoted by bold symbols (e.g., $\mathbf{A}$) and vectors by non-bold symbols (e.g., a). The scalar entry of matrix $\mathbf{A}$ in row i and column j is denoted $\mathbf{A}_{ij}$; similarly, the scalar ith entry of vector a is denoted a_i. Further, the quantities i, j, and k index N_C, N_R, and N_S, respectively (i.e., $i = 1, \ldots, N_C$, $j = 1, \ldots, N_R$, and $k = 1, \ldots, N_S$). For the set of species $\{\mathcal{S}_1, \ldots, \mathcal{S}_{N_S}\}$, a general chemical mechanism is represented as

$$\sum_{k=1}^{N_S} \boldsymbol{\nu}'_{kj} \mathcal{S}_k \rightleftharpoons \sum_{k=1}^{N_S} \boldsymbol{\nu}''_{kj} \mathcal{S}_k, \quad j = 1, \ldots, N_R, \tag{B.1}$$

where $\boldsymbol{\nu}' \in \mathbb{R}^{N_S \times N_R}$ (respectively, $\boldsymbol{\nu}''$) is the reactant (respectively, product) stoichiometric coefficient matrix and $\boldsymbol{\nu} = \boldsymbol{\nu}'' - \boldsymbol{\nu}'$. The formulations below proceed, without loss of generality, in the context that all N_R reactions are reversible. In practice, as discussed in greater detail in Sec. B.1.3, this may not be the case.

The molar net production rate (kmol/m^{3}s) for species k in cell i is

$$\boldsymbol{\Omega}_{ik} = \sum_{j=1}^{N_R} \boldsymbol{\nu}_{kj} \mathbf{Q}_{net_{ij}}, \tag{B.2}$$

where $\boldsymbol{\Omega} \in \mathbb{R}^{N_C \times N_S}$ contains the source terms and $\mathbf{Q}_{net} \in \mathbb{R}^{N_C \times N_R}$ contains the net reaction rates. Note that Eq. B.2 can be expressed concisely through the matrix multiplication $\boldsymbol{\Omega} = \mathbf{Q}_{net}\boldsymbol{\nu}^T$. The complexity comes from the net reaction rate, which is expressed as

$$\mathbf{Q}_{net_{ij}} = \mathbf{Q}_{f_{ij}} - \mathbf{Q}_{r_{ij}} = \mathbf{K}_{f_{ij}} \prod_{k=1}^{N_S} \mathbf{C}_{ik}^{\nu'_{kj}} - \mathbf{K}_{r_{ij}} \prod_{k=1}^{N_S} \mathbf{C}_{ik}^{\nu''_{kj}}. \tag{B.3}$$

Above, $\mathbf{Q}_f$ and $\mathbf{Q}_r \in \mathbb{R}^{N_C \times N_R}$ are the forward and reverse reaction rate matrices, respectively; $\mathbf{K}_f$ and $\mathbf{K}_r \in \mathbb{R}^{N_C \times N_R}$ are the forward and reverse rate constants, respectively; and $\mathbf{C} \in \mathbb{R}^{N_C \times N_S}$ contains the species molar concentrations. Since $\mathbf{Q}_f$ and $\mathbf{Q}_r$ are non-negative, Eq. B.3 can be interpreted as a summation of two ANN layers by enabling matrix multiplications in the logarithm space:

$$\mathbf{Q}_{net} = \exp\big(\log(\mathbf{C})\boldsymbol{\nu}' + \log(\mathbf{K}_f)\big) - \exp\big(\log(\mathbf{C})\boldsymbol{\nu}'' + \log(\mathbf{K}_r)\big). \tag{B.4}$$

It can be seen through Eq. B.4 that the forward and reverse contributions are ANN layers with exponential activation functions, where the input is the logarithm of the concentration matrix $\mathbf{C}$, the weight matrices are known stoichiometric coefficients $\boldsymbol{\nu}'$ and $\boldsymbol{\nu}''$, and the biases are the logarithms of rate constants $\mathbf{K}_f$ and $\mathbf{K}_r$ for the forward and reverse contributions, respectively. These rate constant bias terms can also be interpreted as neural network layers and are the subjects of discussion further below.

Fig. B.3a summarizes the above formulation (Eqs. B.2 and B.4) through an ANN architecture. Note that the leading matrix dimension of all input and output variables, which constitutes the batch size in the forward pass, is N_C. This allows for

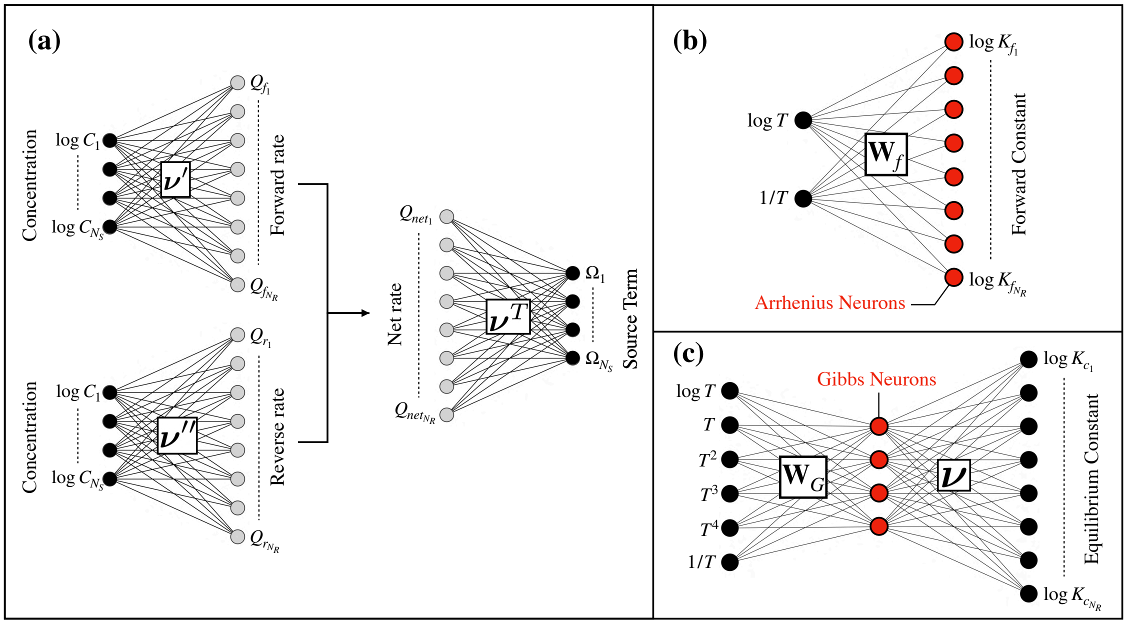


Figure B.3: Illustrations of ANN-based formulations for $N_C = 1$, $N_S = 4$, and $N_R = 8$. Since $N_C = 1$, input/outputs are vectors and cell indices are ignored. (**a**) Schematic of Eqs. B.2 and B.4. Exponential activation functions are used to produce forward/reverse rates. (**b**) Schematic of Arrhenius layer for forward rate constant (Eq. B.6), which is interpreted as a bias term for the output of the forward rate layer in (**a**) (see Eq. B.4). (**c**) Schematic of Gibbs layer equilibrium constant (Eq. B.11). The schematics in both (**b**,**c**) produce the logarithm of the reverse rate constant, which is interpreted as a bias term for the output of the reverse rate layer in (**a**) (see Eq. B.4).

efficient threading and fast execution in high fidelity settings, assuming optimized linear algebra libraries (such as cuBLAS) are utilized by the user. The remaining task, described below, is to obtain the rate constants $\mathbf{K}_f$ and $\mathbf{K}_r$.

The forward rate constant $\mathbf{K}_f \in \mathbb{R}^{N_C \times N_R}$ is given by the Arrhenius expression

$$\mathbf{K}_{f_{ij}} = A_j T_i^{\beta_j} \exp\left(-\frac{E_j}{RT_i}\right), \tag{B.5}$$

where A, β, and E are vectors each of size N_R containing pre-exponential factors, temperature exponents, and activation energies respectively for the elementary reactions. These Arrhenius parameters are known to the user through the mechanism files. The natural logarithm of the forward rate (required in Eq. B.4) usefully yields a form that can also be interpreted as a linear ANN layer,

$$\log(\mathbf{K}_f) = \mathbf{X}_f \mathbf{W}_f + B_f, \text{ where} \tag{B.6}$$

$$\mathbf{X}_f = \begin{bmatrix} \log T_1 & 1/T_1 \\ \log T_2 & 1/T_2 \\ \vdots & \vdots \\ \log T_{N_C} & 1/T_{N_C} \end{bmatrix}, \quad \mathbf{W}_f = \begin{bmatrix} \beta_1 & \cdots & \beta_{N_R} \\ -E_1/R & \cdots & -E_{N_R}/R \end{bmatrix}, \quad B_f = \begin{bmatrix} \log A_1 \\ \log A_2 \\ \vdots \\ \log A_{N_R} \end{bmatrix}^T.$$

In Eq. B.6, $\mathbf{X}_f \in \mathbb{R}^{N_C \times 2}$ is the temperature-dependent input, $\mathbf{W}_f \in \mathbb{R}^{2 \times N_R}$ is a weight matrix consisting of temperature exponents and activation energies, and $B_f \in \mathbb{R}^{1 \times N_R}$ is a bias term of pre-exponential factors. In this sense, each row of the layer output $\log(\mathbf{K}_f)$ can be interpreted as a set of N_R Arrhenius neurons.

The reverse rate constant $\mathbf{K}_r \in \mathbb{R}^{N_C \times N_R}$ is given by

$$\mathbf{K}_{r_{ij}} = \mathbf{K}_{f_{ij}} / \mathbf{K}_{c_{ij}}, \tag{B.7}$$

where $\mathbf{K}_c \in \mathbb{R}^{N_C \times N_R}$ contains the equilibrium rate constants. Since the expression for $\log(\mathbf{K}_f)$ is provided through the Arrhenius neurons (Eq. B.6), the task of determining $\log(\mathbf{K}_r)$ required in Eq. B.4 is accomplished by considering only $\log(\mathbf{K}_c)$.

The equilibrium constant for cell i and reaction j is [261]

$$\mathbf{K}_{c_{ij}} = \left(\frac{p_{ref}}{RT_i} \right)^{\sum_k \boldsymbol{\nu}_{kj}} \exp\left(\frac{\Delta S_j(T_i)}{R} - \frac{\Delta H_j(T_i)}{RT_i} \right), \tag{B.8}$$

where ΔS_j and ΔH_j are changes in entropy and enthalpy for reaction j, and p_{ref} is the reference pressure (1 bar). The logarithm of Eq. B.8 yields

$$\log(\mathbf{K}_{c_{ij}}) = \sum_{k=1}^{N_S} \boldsymbol{\nu}_{jk} \left(-\mathbf{G}_{ik} + \frac{p_{ref}}{RT_i} \right), \tag{B.9}$$

where $\mathbf{G} \in \mathbb{R}^{N_C \times N_S}$ is the nondimensional Gibbs free energy matrix (hereafter referred to as the Gibbs matrix) obtained from the nondimensional enthalpy ($\mathbf{H} \in \mathbb{R}^{N_C \times N_S}$) and entropy ($\mathbf{S} \in \mathbb{R}^{N_C \times N_S}$) matrices. Each entry in the Gibbs matrix is determined from NASA polynomials which provide species enthalpy and entropy as tabulated functions of temperature. The result can be expressed as a matrix multiplication

$$\mathbf{G} = \mathbf{H} - \mathbf{S} = \mathbf{X}_G \mathbf{W}_G + B_G, \text{ where} \tag{B.10}$$

$$\mathbf{X}_G = \begin{bmatrix} \log T_1 & T_1 & T_1^2 & T_1^3 & T_1^4 & 1/T_1 \\ \log T_2 & T_2 & T_2^2 & T_2^3 & T_2^4 & 1/T_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \log T_{N_C} & T_{N_C} & T_{N_C}^2 & T_{N_C}^3 & T_{N_C}^4 & 1/T_{N_C} \end{bmatrix}, \quad \mathbf{W}_G = \begin{bmatrix} \alpha_{1,1} & \dots & \alpha_{1,N_S} \\ \alpha_{2,1} & \dots & \alpha_{2,N_S} \\ \vdots & \ddots & \vdots \\ \alpha_{6,1} & \dots & \alpha_{6,N_S} \end{bmatrix}, \quad B_G = \begin{bmatrix} \alpha_{7,1} \\ \alpha_{7,2} \\ \vdots \\ \alpha_{7,N_S} \end{bmatrix}^T.$$

In Eq. B.10, $\mathbf{X}_G \in \mathbb{R}^{N_C \times 6}$ is the input consisting of various functions of temperature and $\boldsymbol{\alpha} \in \mathbb{R}^{7 \times N_S}$ is a matrix of polynomial coefficients; the first six rows of $\boldsymbol{\alpha}$ is the weight matrix $\mathbf{W}_G$ and the last row is the bias B_G. Note that, although not shown in Eq. B.10 for conciseness, the quantities in $\boldsymbol{\alpha}$ (and in turn $\mathbf{W}_G$ and B_G) are also functions of the cell temperature T_i and the species index. This is because the species polynomial coefficients change based on a cutoff temperature (usually 1000 K). Regarding practical implementation, in the case that the NASA cutoff temperature is the same for all of the species (say 1000 K), it is possible to treat the evaluation of Eq. B.10 (the first layer in Fig. B.3c) with linear algebra libraries (i.e., cuBLAS operations) by populating two copies of $\boldsymbol{\alpha}$ based on the cutoff threshold. However, this approach is only feasible in the case when the NASA cutoff temperature is the same for all of the species. In the more general case where the cutoff temperature varies with the species index, it is much more convenient to directly treat the evaluation of Eq. B.10 using a custom matrix multiplication kernel that computes the polynomial directly—with conditional logic that populates the values in $\boldsymbol{\alpha}$ on-the-fly based on the cell temperature—instead of a using the cuBLAS (or other similar) routines. The convenience here outweighs the hit on GPU optimality, and, as such, this is the approach used in the GPU implementation profiled in Sec. B.2.

Inserting Eq. B.10 into Eq. B.9 gives

$$\log(\mathbf{K}_c) = -(\mathbf{X}_G \mathbf{W}_G + B_G)\boldsymbol{\nu}, \tag{B.11}$$

where the standard concentration term p_{ref}/RT_i has been integrated into the bias B_G. Eq. B.11 can be interpreted as a linear two-layer ANN. The parameters of the first layer (the Gibbs layer) are the temperature-dependent $\mathbf{W}_G$ and B_G and those of the second layer are the net stoichiometric coefficients $\boldsymbol{\nu}$. The intermediary neurons (i.e., hidden layer neurons) here are referred to as the Gibbs neurons.

Illustrations of both the forward and the equilibrium rate constant formulations as neural network-inspired architectures are shown in Fig. B.3b,c, with Arrhenius and Gibbs neurons highlighted. Alongside the formulations provided above, however, some additional complications arise when considering the forward rate constant in detailed reaction mechanisms and the treatment of zero concentrations. These are considered in the present work and brief descriptions on their implementation are described below.

To handle three-body reactions, the quantity $\log(\mathbf{M})$ is added to $\log(\mathbf{K}_f)$, where $\mathbf{M} \in \mathbb{R}^{N_C \times N_R}$ is a matrix of third-body concentrations for each reaction ($\mathbf{M}_{ij}$ is 1 if reaction j does not include a third body). The entries in $\mathbf{M}$ can be obtained through the matrix multiplication $\mathbf{M} = \mathbf{CE}$, where $\mathbf{E} \in \mathbb{R}^{N_S \times N_R}$ is a matrix of third-body efficiency factors. In practice, for computational savings, the third body concentration matrix equation is evaluated only for the subset of total reactions N_R, since third body concentration terms are generally not present in every single reaction.

Additionally, falloff and pressure-log reactions, which constitute two of the reaction types discussed further below in Sec. B.1.2, are treated separately from standard Arrhenius reactions as defined in Eq. B.6. In summary, they modify the standard matrix-based formulation of the forward rate constant logarithm by incorporating much more complex and arithmetically intensive expressions. As such, pressure-log and falloff formulations such as that of Lindemann, Troe, or SRI are handled on the GPU in a custom non-matrix fashion. More information on the mathematical implementation expressions for these reactions can be found in the Cantera and Chemkin documentation [109, 144].

Lastly, it is necessary to acknowledge that the treatment of kinetics in the logarithm space introduces scenarios in which logarithms of zero concentration are required. There are several ways to treat this with effectively zero propagating error – some options are listed below.

1. Let the GPU handle the zero concentrations natively. For example, CUDA supports the operation of `exp(log(0))=0` through the `Inf` floating point placeholder. That is, CUDA ensures that the operation `log(0)=-Inf`, and that `exp(-Inf)=0`.

2. Replace either the zero concentrations or mass fractions with an extremely small positive number and then proceed with the computations. This number should be small enough (i.e. `1.0e-300`) to effectively produce a huge negative number upon taking the logarithm.

3. Replace the logarithm of the zero concentrations directly with a huge negative

number, e.g., through a re-definition as `log(0)=-1.0e300`.

It should be noted that errors are indeed introduced from all of the above treatments of the zero logarithm, as it is mathematically undefined and the numerical treatments incur some truncation penalties. However, these errors are microscopic relative to other error sources that are encountered in a reacting flow simulation and are unnoticeable in practice (verification is provided in the Appendix of *Barwey and Raman* [16]).

B.1.2 Organization of Data

The notion of N_C, N_R, and N_S introduces three matrix types that represent the backbone of the methodology discussed above: an $N_C \times N_R$ reaction matrix, an $N_C \times N_S$ species matrix, and an $N_S \times N_R$ (or the transpose) stoichiometric matrix. As per the formulations in Sec. B.1.1, reaction matrices are used to define quantities such as rate constants and reaction rates, whereas species matrices are used to define concentrations, mass fractions and species net production rates. Stoichiometric matrices can be interpreted as tools that transform a matrix from the species representation to the reaction representation or vice versa.

Assuming the matrices occupy contiguous blocks of memory, there are two ways to go about their storage: a row-major or a column-major representation. As an example, consider some given $N_C \times N_R$ matrix $\mathbf{A}$ (a reaction matrix). In the row-major format, the reaction matrix is represented as an N_C-sized stack of $1 \times N_R$ row vectors. On the other hand, in the column-major format, the matrix is interpreted as an N_R-sized stack of $N_C \times 1$ column vectors. This difference is shown in Fig. B.4a. There are a few reasons the column-major representation is more beneficial here. First, in most realistic high-fidelity applications, we have the condition $N_C \gg N_R > N_S$. From a GPU efficiency perspective, it is important to have memory coalescence along

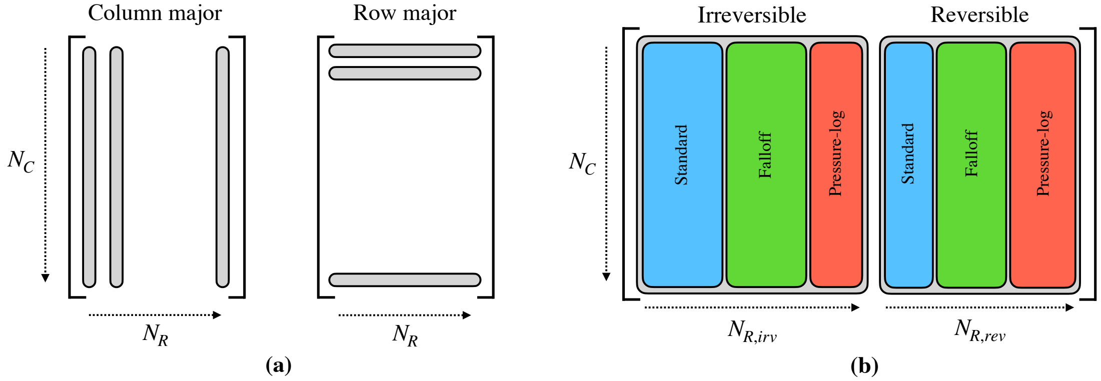


Figure B.4: (**a**) Interpretations of a matrix in column major (**left**) and row major (**right**) formats. The gray shapes within the matrices show the storage method as a stacking of N_R column vectors (for column major) or N_C row vectors (for row major). (**b**) Decomposition of a reaction matrix into $N_{R,irv}$ irreversible and $N_{R,rev}$ reversible sub-matrices, each with another set of sub-matrices corresponding to standard, falloff, and pressure-log reaction types.

the highest dimension N_C, which necessitates a column-major storage format for $\mathbf{A}$.[1] Second, the practical implications of designing a matrix as a collection of $N_C \times 1$ vectors is appealing – it allows for naturally extracting contiguous $N_C \times N_X$ subsets of the $\mathbf{A}$ matrix, where $N_X \leq N_R$. This becomes especially useful when dealing with different reaction types as discussed in Sec. B.1.3. Lastly, the column-major format allows for seamless integration with cuBLAS, a highly efficient GPU-based linear algebra library developed by NVIDIA.

B.1.3 Reaction Decomposition and Classification

As mentioned in Sec. B.1.1, not all reactions need to be reversible in a given mechanism. Oftentimes, Arrhenius parameters for *reverse* rate constants are provided directly, in which case a physically reversible reaction is supplied to the user through the mechanism files as two irreversible (forward) reactions parameterized by

[1] Good coalescence ensures that, for a given number of floating point operations (FLOPs) in the kernel, the FLOPs executed by the GPU per byte of transferred global memory (arithmetic intensity) is as high as possible. Arithmetic intensity is used as a general metric to assess GPU performance, and it is discussed further in Sec. B.2.1. For more information on the specifics of global memory coalescence, the reader is referred to the CUDA toolkit documentation.

the corresponding Arrhenius constants. In this scenario, the computation of the equilibrium rate constant (Eq. B.9) for one reaction is effectively traded for an additional Arrhenius rate constant evaluation (Eq. B.6).

With this in mind, any given reaction matrix $\mathbf{A}$ initialized as a contiguous column-major matrix of size $N_C \times N_R$ is constructed such that it decomposes into two sub-matrices: one for *irreversible* reactions and the other for *reversible* reactions. More formally, in general, there are $N_{R,irv}$ and $N_{R,rev}$ irreversible and reversible reactions, respectively, where $N_R = N_{R,irv} + N_{R,rev}$. The matrix $\mathbf{A}$ is then partitioned into two smaller matrices along the reaction axis as $\mathbf{A} = [\mathbf{A}_{irv}; \mathbf{A}_{rev}]$, where $\mathbf{A}_{irv}$ is the $N_C \times N_{R,irv}$ sub-matrix of $\mathbf{A}$ corresponding to irreversible reactions, $\mathbf{A}_{rev}$ is the $N_C \times N_{R,rev}$ sub-matrix corresponding to reversible reactions, and $[;]$ is a concatenation operator. This separation is advantageous in practice because additional matrix operations are required for reversible reactions (i.e., the equilibrium rate constant evaluations of Eq. B.9) that are absent from irreversible reactions. Since these operations are expensive, performing them on the full matrix $\mathbf{A}$ is wasteful for mechanisms that observe a large number of irreversible reactions. Furthermore, the column-major data structure ensures that the two sub-matrices $\mathbf{A}_{irv}$ and $\mathbf{A}_{rev}$ are still contiguous in memory, a necessary requirement for GPU-optimized matrix multiplication algorithms. As a result, the arrangement of a reaction matrix as two concatenated sub-matrices allows for efficient matrix operations on both subsets, or the full matrix, whenever necessary.

In a similar process, each sub-matrix $\mathbf{A}_{irv}$ and $\mathbf{A}_{rev}$ can again be decomposed into a set of smaller sub-matrices categorized by specific reaction types. As mentioned above, in this work, the three most commonly encountered reaction types are considered:

1. *Standard reactions* utilize the standard Arrhenius equation for the forward rate constant (Eq. B.6). They may or may not include third body contributions.

2. *Falloff reactions* employ significantly more complex expressions of the forward rate constant computation for reactions involving third bodies. Falloff reactions can be Lindemann, Troe, or SRI type, each with slightly different implementations. The reader is referred to the Cantera [109] or Chemkin [144] documentation for mathematical details.

3. *Pressure-log reactions* add pressure dependence into the computation of the forward rate constant through an interpolation routine. They may or may not include third body contributions. The reader is referred to the Cantera or Chemkin documentation for mathematical details.

In the most general case, for each set of reversible and irreversible reactions, the decomposition into reaction types ensures

$$N_{R,irv} = N^S_{R,irv} + N^F_{R,irv} + N^P_{R,irv}, \tag{B.12}$$

$$N_{R,rev} = N^S_{R,rev} + N^F_{R,rev} + N^P_{R,rev}, \tag{B.13}$$

where the superscripts S, F, and P indicate standard, falloff, and pressure-log reactions respectively (e.g., $N^S_{R,irv}$ is the number of irreversible, standard-type reactions in the mechanism). Additionally, the total number of reaction types present in the mechanism discounting reaction reversibility is given by

$$N^S_R = N^S_{R,irv} + N^S_{R,rev}, \tag{B.14}$$

$$N^F_R = N^F_{R,irv} + N^F_{R,rev}, \tag{B.15}$$

$$N^P_R = N^P_{R,irv} + N^P_{R,rev}, \tag{B.16}$$

such that $N_R = N^S_R + N^F_R + N^P_R = N_{R,irv} + N_{R,rev}$. In practice, for memory coalescence reasons, the matrices $\mathbf{A}_{irv}$ and $\mathbf{A}_{rev}$ are again partitioned in a similar manner as described above, creating a two-level hierarchy. The top-most level, to reiterate,

Class	Reference	Description	N_S	N_R
A1	Mueller et al. [212]	Hydrogen/Air	9	21
A2	FFCMy-12 [319, 359]	Methane/Oxygen	12	38
B1	FFCMy-30 [359]	Ethylene/Air	30	231
B2	UCSD [232]	Hydrogen/Air	57	268
B3	FFCMy-40 [359]	Ethylene/Air	41	361
C1	AramcoMech 1.3 [204]	–	253	1542
C2	LLNL [201]	–	654	4846

Table B.1: List of mechanisms used throughout this work arranged in ascending order of N_R. They are grouped into three classes based on N_R for ease of reference. Class A is for smaller mechanisms ($N_R = \mathcal{O}(10)$), Class B for medium-sized mechanisms ($N_R = \mathcal{O}(100)$), and Class C for large mechanisms ($N_R = \mathcal{O}(1000)$). Note that the mechanism labeled C1 is unrelated to the jet fuel of the same name.

is $\mathbf{A} = [\mathbf{A}_{irv}; \mathbf{A}_{rev}]$. The second level then provides $\mathbf{A}_{irv} = [\mathbf{A}^S_{irv}; \mathbf{A}^F_{irv}; \mathbf{A}^P_{irv}]$ and $\mathbf{A}_{rev} = [\mathbf{A}^S_{rev}; \mathbf{A}^F_{rev}; \mathbf{A}^P_{rev}]$. A schematic of this decomposition is provided Fig. B.4b.

For a given mechanism, the above decomposition (or classification) into reaction types applies to all reaction matrices, i.e., matrices that are of size $N_C \times N_R$. Note that this is a by design an overly general classification. Most mechanisms do not contain every aforementioned reaction type. For example, one mechanism may only contain reversible reactions ($N_{R,irv} = 0$), and another may contain both reversible and irreversible reactions, but no falloff reactions ($N^F_R = 0$). Each chemical mechanism can therefore be characterized not only by the total number of reactions, but also by the distribution of the reaction types. The mechanisms considered in this work are detailed in Table B.1, and their corresponding reaction type distributions are visualized in Fig. B.5. Sec. B.2 shows how knowledge of these distributions gives additional insight into the speedup and GPU performance behavior, since each reaction type brings forward a different amount of numerical complexity.

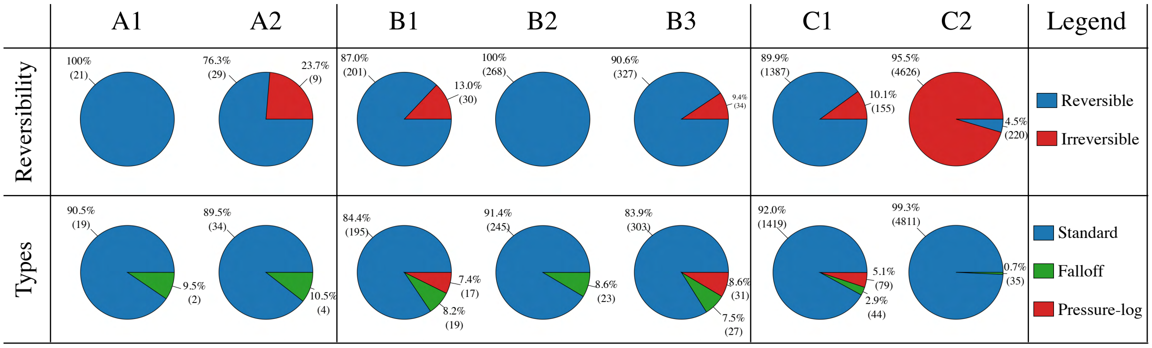


Figure B.5: Reaction distributions shown as pie charts for the mechanisms listed in Table B.1. The top row shows proportions of reversible and irreversible reactions, whereas the bottom row shows proportions of standard, falloff, and pressure-log reactions. Indicated for each wedge is the proportion (reaction number normalized by total number of reactions) as a percentage, and the absolute reaction number in parentheses. The significant difference in the reaction distribution for C2 with respect to reversibility is due to the fact that most of the reversible reactions were parameterized with Arrhenius expressions for both forward and reverse components (see the introductory comments of Sec. B.1.3).

B.2 GPU Performance Analysis

This section characterizes the GPU performance trends derived from the matrix-oriented methodology described in Sec. B.1. In particular, speedup and saturation behaviors are assessed with respect to: (a) the chemical mechanisms in Table B.1; (b) the leading matrix dimension N_C (here interpreted as the number of reacting cells in a computational domain assigned to one GPU); and (c) the reaction distributions provided in Fig. B.5. The performance analysis is based on trends for absolute compute times and throughput for the various arithmetic operations encountered during the source term computation. Cost breakdowns for different reaction types and specific advantages of the matrix-based formulation in light of mechanism sparsity properties are also explored. The goal is ultimately to provide insight into how the computationally intensive source term calculations scale with the reaction mechanism complexity (Sections B.2.1 and B.2.2), which types of reactions benefit the GPU formulations most (Sec. B.2.3), and how to exploit the matrix-based formulations to provide optimal speedup for large mechanisms (Sec. B.2.4).

GPU performance in this work is assessed purely from the perspective of the source term computation in isolation. The GPU-enabled speedup for an entire reacting flow solver can drastically vary depending on: (a) the chemistry time-integration algorithm; (b) GPU treatment of convective/diffusive fluxes; (c) GPU treatment of boundary conditions and domain decomposition communication steps; and (d) the amount (and implementation of) CPU-GPU data transfers. Since the methodology in Sec. B.1 exists independently from these factors, the GPU speedup and performance trends are also treated independently. Details on full solver integration using the matrix-based kinetics methodology are found in Refs. [30, 340].

The methodology described in Sec. B.1 was implemented on the GPU with a C++ library, `UMChemGPU`, serving as a high-level API to call lower-level CUDA and cuBLAS routines. The API is a part of a larger GPU-based library used for offloading computationally intensive routines commonly found in high-fidelity unstructured multi-physics solvers; details of the full library will be provided in a future manuscript. Further, since the formulations presented in Sec. B.1.1 are exact, a meticulous verification study for the GPU implementation is not included here.

The calculations used in the analysis below were performed in double precision on an ORNL Summit node consisting of IBM Power9 CPUs and NVIDIA V100 GPUs. All performance-related quantities were obtained from the `nvprof` profiling tool. When constructing the performance profiles, species mass fractions were obtained arbitrarily using a random sampling procedure and a normalization step to ensure sums of unity—the choice of mass fractions (and species concentration) has no effect on the profiling results, as the algorithm arithmetic and global memory read/write points are independent of the species mass fraction distributions. Lastly, it should be noted that absolute values of speedup will of course depend on the node hardware as well as the user implementation of GPU functions—for these reasons, the saturation and throughput trends (i.e., a measure of how well the GPU resources are being utilized with respect to the theoretical limits) are more valuable overall, as they better isolate GPU performance dependence on mechanism complexity over compute architecture.

B.2.1 Compute Times and Throughput

Absolute GPU compute times for source term evaluation as a function of the leading matrix dimension (or number of cells) N_C are shown in Fig. B.6. Each curve is characterized by three sequential features present in all GPU-based profiles: the pre-saturated regime, the saturation point, and the saturated regime. In the pre-saturated regime, the compute time curve observes a near-zero slope with increasing computational complexity, where the effective knob for computational complexity for a given reaction mechanism is here available through N_C. In other words, in the pre-saturated regime, the compute cost is independent of N_C, implying that the GPU resources are not fully utilized. On the other hand, the saturated regime is characterized by a "filling-up" of the GPU resources where the compute time increases roughly linearly with increasing N_C. Finally, the *saturation point* is the marker that indicates the onset of saturated regime, i.e., the point N_C at which the curve roughly begins to adhere to the linear trend.

In light of these three features, Fig. B.6 reveals many useful trends with regards to reaction mechanism complexity. Perhaps the most apparent is the expected behavior of increased GPU compute time with increasing mechanism complexity for any given N_C—in other words, as one traverses from mechanism A1 to C2 with N_C fixed, the compute time increases. Another more subtle trend lies in the N_C-locations of the indicated saturation points. Namely, the saturation point is correlated almost uniquely with the number of reactions, N_R, as opposed to the number of species, N_S. As N_R increases by an order of magnitude (as one moves from mechanism class A to B to C), the saturation point decreases in N_C by roughly the same order of

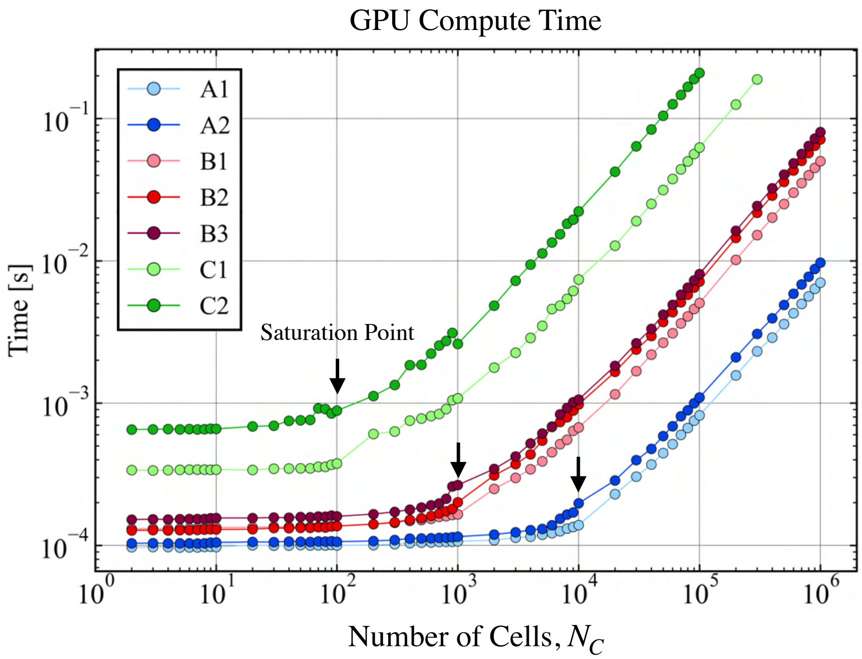


Figure B.6: Absolute compute time (time-to-solution) as a function of cell number N_C for the GPU evaluation of the species source terms for the mechanisms listed in Table B.1. Saturation points for each mechanism group are indicated by the black arrows.

magnitude factor. A similar trend is seen in the saturated regime, where, for a given N_C (say 10^5), the average compute time for each mechanism class increases by order of magnitude intervals, i.e., in proportion to the increase in N_R. Interestingly, the N_R-based correlation is absent in the pre-saturated regime; the general trend of increased compute time is indeed present, but, prior to saturation, increases in compute time are not proportional to same degree of increase in N_R. This behavior is characterized the dominance of the computational budget in the saturated regime by dense matrix multiplications involving very large reaction matrices, and is discussed in more detail in Sec. B.2.4 to motivate the usage of sparse algorithms.

The information obtained from raw compute times in Fig. B.6 becomes much more valuable when contextualized with a direct representation of GPU performance relative to the theoretical limits of the hardware in question. This is the primary purpose of the Roofline model [354], which is the de-facto judge in the GPU computing literature for assessing saturation, GPU utilization, and throughput behavior for any given application. Figure B.7a shows a typical Roofline plot. Before delving into the details of the figure, the basics of the model are first described. The reader already familiar with Rooflines may skip the next two paragraphs.

Briefly, there are two primary goals of the Roofline model. The first is to assess whether or not a GPU kernel (or function) is bandwidth-limited or compute-limited, and the second is to inform whether or not the theoretical limits of the hardware have been reached by the kernel. The Roofline model illuminates both of these goals in a single plot through the *arithmetic intensity* (x-axis of Fig. B.7) and *throughput performance* (y-axis). For a given GPU kernel, arithmetic intensity is the ratio of floating point operations (FLOPs), as defined by the kernel arithmetic, to the amount of data (in bytes) transferred to and from the global memory source, as defined by the kernel inputs/outputs. On the other hand, throughput performance is the ratio of kernel FLOPs to execution time in seconds. In summary, arithmetic intensity has units FLOPs-per-byte and throughput performance has units of FLOPs-per-second (typically, the performance is given in units of GigaFLOPs-per-second).

Theoretical peak performance of the GPU (the horizontal black line in Fig. B.7a) is available only when kernels have sufficiently high arithmetic intensity; otherwise, the kernels are bandwidth-limited. The cutoff point at which a kernel goes from bandwidth to compute-limited is the Roofline elbow—for arithmetic intensities below the elbow, the maximum achievable performance decreases at a linear rate defined by the hardware (typically measured in Gigabytes-per-second). The effective GPU utilization for a kernel can then be visualized directly by plotting the two attributes of the kernel (throughput performance versus arithmetic intensity) and comparing with the theoretical device limits. Ideally, all kernels should approach the throughput limits of the hardware regardless of arithmetic intensity. Lastly, it should be noted

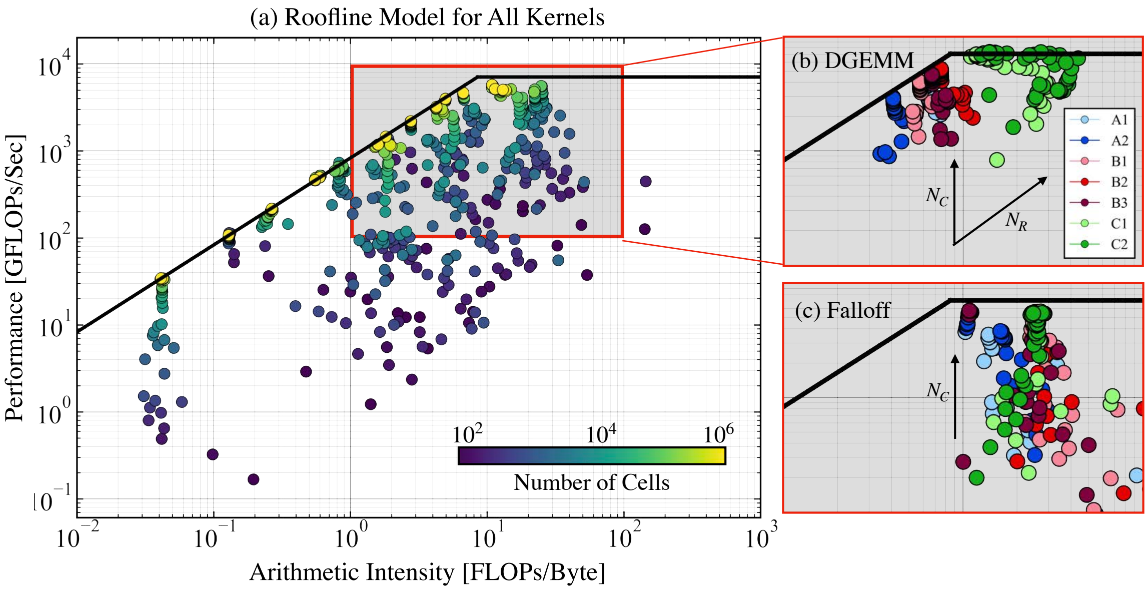


Figure B.7: (**a**) Roofline model for mechanism B3 for all kernels encountered during the source term computation. Each marker represents a unique kernel evaluated for the number of cells N_C indicated by its color. (**b**) Roofline model for the DGEMM kernel. (**c**) Roofline model for the Troe falloff kernel. In (**b**,**c**), the different colors represent different mechanisms (see Table B.1) and the distribution of points of the same color comes from the various N_C values.

that the Roofline model concerns rates and not absolute quantities (i.e., throughput performance and execution time are not interchangeable). In other words, kernels that are compute-limited do not necessarily run faster in physical time than those that are bandwidth-limited.

Proceeding with the analysis, Fig. B.7a shows a Roofline plot *for all GPU kernels* encountered during the source term evaluation using different N_C values for a single chemical mechanism (B3) for brevity, as global Roofline trends with respect to N_C for all mechanisms are similar. Each point represents a kernel (i.e., matrix multiplication operation, Gibbs matrix computation, exponentiation, etc.), and the color of the point in Fig. B.7a denotes the N_C for which the kernel was evaluated. It is apparent that: (a) most of the kernels are bandwidth-limited; and (b) the kernels approach near-theoretical limits in all cases as N_C reaches the saturation point. In general, increasing the value N_C moves a kernel defined for a particular arithmetic intensity vertically on the Roofline plot, eventually saturating the GPU and achieving theoretical efficiency. Thus, for a large-scale application, one should allocate the MPI resources to accommodate a high enough N_C per GPU in order to ensure that node resources are utilized to their fullest extent.

Examples of Rooflines extracted for *individual kernels* are shown in Fig. B.7b,c. In these figures, the points are colored by reaction mechanism instead of N_C to assess direct effects of mechanism complexity. The spread of points for a single mechanism (a single color) represents the various N_C values at which the kernel was computed.

Figure B.7b shows the progression of the double-precision general matrix multiplication (DGEMM) operation provided by the cuBLAS backend, which is the main driver for the matrix-based formulation implemented in this work. Immediately apparent is an increase in arithmetic intensity with N_R (i.e., the points are cleanly clustered by reaction mechanism). The Class C mechanisms in particular allow for the DGEMM operation to access peak performance for most N_C values. The smaller mechanisms are near the tail-end of the bandwidth-limited region, though they still achieve theoretical peaks for sufficiently high N_C.

Figure B.7c shows analogous results for the Troe falloff kernel, which is one of the most arithmetically intensive routines encountered during source term computation. The Troe falloff kernel results of Fig. B.7c also represent the results for other similarly complex kernels such as Lindemann/SRI falloffs, Pressure-log reactions, and Gibbs free energy evaluations. Interestingly, the mechanism-based clusters seen for DGEMM are entirely absent here. As per the complexity of the kernel, all mechanisms lie in the compute-limited region and reach near peak performance. The consequence of the complexity of the Troe falloff kernel, however, lies in the high amount of variance in performance for a given mechanism (spread in the y-axis) as compared to the simpler, more optimized DGEMM routine. This is especially evident for mechanism C2: the complexity of the Falloff function has effectively traded the horizontal spread at peak performance seen in DGEMM (desired behavior) with vertical spread at near-peak performance (less desired). A primary cause for this behavior is implied by reaction distributions of Fig. B.5. Since, for a given mechanism, the DGEMM kernel is utilized on a higher proportion of the total number of reactions than the falloff kernel (this is evidenced by the standard versus falloff percentages in the reaction type distributions), the theoretical device limits are expected to be achieved for a wider range of N_C for the DGEMM kernel. Additional causes include both the lack of GPU-optimality of the form of the analytic falloff functions itself, and the fact that the falloff kernels must be implemented as custom CUDA routines; that is, the advantages of using a highly optimized backend such as cuBLAS are simply less fruitful for a mechanism containing more falloff (and other similarly complex) reactions or functions.

B.2.2 Speedup

The corresponding GPU speedup curves are shown in Fig. B.8 in both logarithmic (left) and linear (right) scales. The CPU baseline from which the speedup is derived is the C++ Cantera function `getNetProductionRates` evaluated with one MPI rank. The speedup is intended to capture the tangible offloading effect for a single MPI rank (or OpenMP thread) in the common case of a one-to-one correspondence between MPI process and GPU. Although the Cantera CPU baseline does not present the fairest comparison here (the arithmetic operations in Cantera are not vectorized in the same way as those defined in Sec. B.1.1), it is routinely used in numerical reacting flow simulations for kinetics and thermodynamics routines. As such, using this baseline to derive the GPU speedup provides valuable information on the practical impact of a CPU-to-GPU offload for many existing applications that use Cantera or comparable libraries such as Chemkin. Although the CPU reference values used to compute the speedup in Fig. B.8 come from the Cantera functions, the matrix-based forms were also evaluated on the CPU to assess the effect of the Cantera implementations on the perception of GPU speedup. It was found that for the same mechanisms, the matrix-based formulations implemented on the CPU provided roughly a factor of 2 speedup over the CPU-based Cantera functions for most cell counts. In other words, the GPU speedup discussed below drops by roughly a factor of 2 when comparing against the matrix-based formulation on the CPU instead of the Cantera function, though the saturation trends remain the same. To reiterate, the speedups themselves are secondary to the relative trends displayed between the individual reaction mechanism curves.

As shown in Fig. B.8, the speedup increases linearly with cell number until the saturated regime is reached, wherein the speedup stagnates at reasonably high values across the board. This general trend is expected since the saturated regime is defined by a GPU compute time that increases linearly with N_C (Fig. B.6). The consequence is that the maximum speedup is reached for lower values of N_C as mechanism complexity increases, as per the location of the saturation point. The converged speedup values are high overall, varying between 100× to 500× at their maximum. However, due to the dependence on the CPU implementation, the mechanism trends with respect to speedup are not as intuitive as those present in the absolute compute time curves of Fig. B.6. For instance, the pre-saturated regime (e.g., $N_C = 10$) shows increasing speedup with mechanism complexity, but this trend is absent in the saturated regime.

Instead, in the saturated regime, the behaviors can be split into two groups. The first group concerns mechanism Classes A and B, which concentrate in the regions of 400–500× speedup. For these mechanisms the speedup trends are related directly to the reaction mechanism distributions seen in Fig. B.5. In other words, for Classes A and B, similar distribution types converge to similar saturated speedups. For example, Mechanisms B1 and B3 converge near 500× and observe near-identical

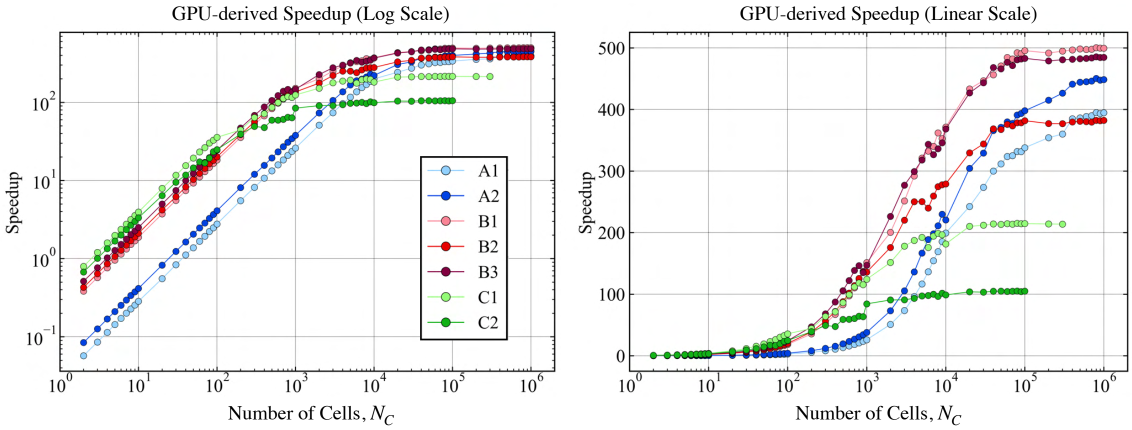


Figure B.8: (**Left**) GPU-derived speedup in log-scale with respect to the Cantera-based CPU baseline for all mechanisms listed in Table B.1. (**Right**) Same as left but with linear-scale in the y-axis.

reaction distributions—the same is true for Mechanisms A1 and B2, which converge near 400×. The implication is that the GPU based speedup relies not only on the number of reactions, but also on reaction types encountered in the mechanism. This is the central focus of Sec. B.2.3.

The second group is tied to the larger mechanisms of Class C, where the aforementioned trends break down. Though still high, the saturated speedups in Fig. B.8 progressively drop from C1 to C2. This drop relates directly to the connection between: (a) the presence of the overbearing reaction matrix ($N_C \times N_R$ matrix) multiplications encountered in Eq. 2.9, which constitute a bottleneck for large mechanisms; and (b) the mechanism sparsity as measured by the number of elements in the matrix $\boldsymbol{\nu}$. The details of the interplay between these two elements, which leads to a significant improvement in the Class C speedups observed in Fig. B.8 by overcoming the mentioned bottlenecks, are postponed to the end of this section (Sec. B.2.4). The discussion hereafter continues in the context of the speedups shown in Fig. B.8 as they stand.

B.2.3 Cost of Reaction Types

The above sections described general speedup and performance trends for the GPU in light of the matrix-oriented methodology of Sec. B.1.1. As per the classifications presented in Sec. B.1.3, a more detailed investigation into the GPU cost of individual reaction types in the pre-saturated and saturated regimes can better illuminate both the sources of speedup behavior and the types of mechanisms that are more suited for GPU offloading.

To better isolate the GPU effects of individual routines and reaction types, the quantity of interest here is defined as the cost per reaction. The cost per reaction is measured by taking the compute time of a specific kernel (or routine composed of a set of kernels) and normalizing by the size of the subset of total reactions on which the routine acts. For example, in the case of reversible reactions, the computation of equilibrium rate constants occur on the subset of total reactions of size $N_{R,rev}$. To get a measure on the total time *per reaction* taken during the computation of $\log(\mathbf{K}_c)$, the corresponding routine time is divided by $N_{R,rev}$. This effectively provides an averaged measure on the cost impact for individual reaction types or routines encountered in a given mechanism.

Figure B.9 compares the cost per reaction for $\log(\mathbf{K}_f)$ and $\log(\mathbf{K}_c)$ routines for the three Mechanisms A2, B3, and C1. The $\log(\mathbf{K}_f)$ routine consists of the following: Arrhenius-based matrix multiplication (Equation (B.9), uses a DGEMM kernel), evaluation of third body concentrations if present (DGEMM kernel), falloff rate constant evaluations if applicable (custom kernels), and pressure-log rate constant evaluations if applicable (custom kernels). The $\log(\mathbf{K}_c)$ routine consists of the evaluation of Equation (B.9), which utilizes a first custom kernel to fill the Gibbs matrix and a second DGEMM kernel to recover the rate constants.

In the pre-saturated regime, the cost per reaction of the $\log(\mathbf{K}_f)$ routine consistently outweighs that of $\log(\mathbf{K}_c)$. However, the behavior is different in the saturated

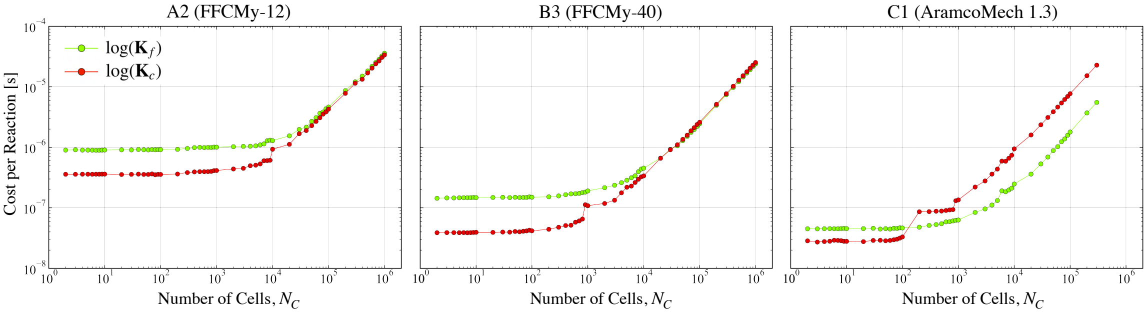


Figure B.9: Cost per reaction for forward rate constant ($\log(\mathbf{K}_f)$) and equilibrium constant ($\log(\mathbf{K}_c)$) routines as a function of N_C for: Mechanism A2 (**left** plot); Mechanism B3 (**middle**); and Mechanism C1 (**right**).

regime as mechanism complexity grows. An interesting feature is the consistent spike in cost per reaction for $\log(\mathbf{K}_c)$ at the saturation points (see Fig. B.6) of the respective mechanisms. For the smaller mechanisms, this jump leads to a convergence in the difference between $\log(\mathbf{K}_f)$ and $\log(\mathbf{K}_c)$ costs in the saturated regime. For the large Mechanism C1, this jump at the commencement of saturation near $N_C = 10^2$ brings the $\log(\mathbf{K}_c)$ cost per reaction to a higher value than the $\log(\mathbf{K}_f)$ cost (though not shown here, this effect is exacerbated for mechanism C2). The implication is that more complex mechanisms (as determined by N_S and N_R) incur a higher penalty for $\log(\mathbf{K}_c)$ evaluations in the saturated regime. As such, in the case of very large mechanisms, a useful driver to design mechanisms that are more "GPU-optimal" in the saturated regime is to minimize the presence of reversible reactions, i.e., to steer the mechanism distribution towards something akin to that of C2 (see the top row of Fig. B.5).

Analogous costs for the three individual reaction types described in Sec. B.1.3 – standard, pressure-log, and falloff – are shown for all mechanisms in Fig. B.10. Recall that these reaction types modify the computation of the forward rate constant whenever applicable (i.e., they are sub-components of the $\log(\mathbf{K}_f)$ computation). The trends can again be characterized by behaviors in the pre-saturated and saturated regimes. For all reaction types, the cost per reaction decreases with respect to

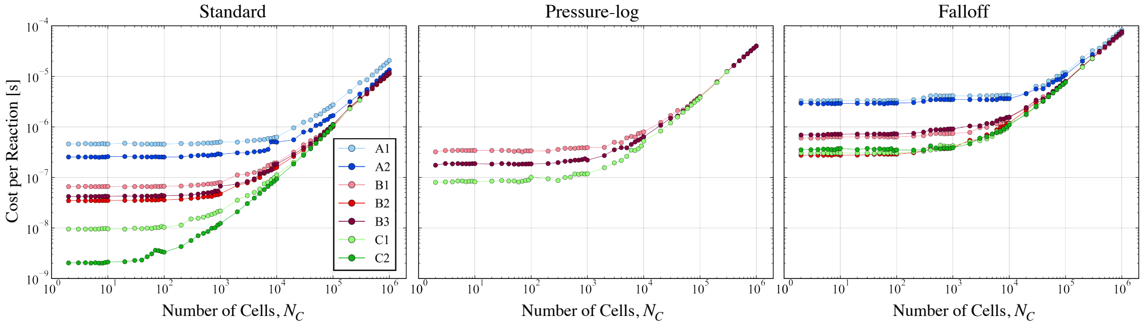


Figure B.10: Cost per reaction for all mechanisms for: standard reactions (**left**); pressure-log reactions (**middle**); and falloff reactions (**right**). See Table B.1 for mechanism information.

mechanism complexity, most notably for the standard reactions. This is likely the cause of the inverse trend in speedup (Fig. B.8) seen in the same pre-saturated regions. Further, when comparing across reaction types, standard reactions (based on the cuBLAS DGEMM) are significantly cheaper than pressure-log and falloff counterparts. This is expected due to the inherent differences in arithmetic complexity and also to the optimized implementations of the cuBLAS backend. In-line with the Roofline-derived findings of Sec. B.2.1, mechanisms that attempt to minimize the presence of non-standard reactions are likely to achieve a greater N_C-range of cost efficiency.

A prominent feature in Fig. B.10 is the convergence, for a specific reaction type, of all mechanism curves to the same cost per reaction in the saturated regime. This implies that, for the reaction types considered during the evaluation of $\log(\mathbf{K}_f)$, the cost advantage due to mechanism complexity *per reaction* only applies in the pre-saturated regime. The fact that the cost per reaction mechanism curves converge here in the saturated regime is consistent with the compute time trends seen earlier in Fig. B.6 – namely, that the overall time-to-solution in the saturated regime increase with each mechanism by the same factor of increase in N_R.

B.2.4 Improving Speedup for Large Mechanisms

A valid concern related to the speedup in Fig. B.8 is the decreasing trend for Class C mechanisms. A procedure to remove this speedup deterioration for these larger mechanisms is provided here. Put briefly, the procedure relies on taking advantage of the inherent sparsity that characterizes mechanisms with high N_S and N_R values. This concept is first motivated by a computational budget analysis of the original routines leading to the speedups shown in Fig. B.8. It is then shown that a simple replacement of several problematic dense matrix multiplications with sparse counterparts very usefully recovers the "lost" speedup for the Class C mechanisms.

The computational budget, measured as the proportion of total compute time spent in the respective routines, is provided in Fig. B.11 for the same three mechanisms shown in Fig. B.9. For each mechanism and N_C, the budget is split into four main components encountered during the source term computation:

1. *The preprocessing routine* consists of kernels that: (a) recover the primitive species mass fractions from conserved values; (b) normalize the species mass fractions in each cell to sum to unity; and (c) obtain molar concentrations from the mass fractions.

2. *The forward rate constant routine* for $\log(\mathbf{K}_f)$ computes Equation (B.6) along with any other non-standard reaction types that modify the forward rate (three-body, pressure-log, and falloff).

3. *The equilibrium rate constant routine* for $\log(\mathbf{K}_c)$ is outlined in Equation (B.9).

4. *The species net production rate routine* for $\mathbf{\Omega}$, given by Equation (B.2)—the cost of evaluating the net reaction rates $\mathbf{Q}_{net}$ required to recover $\mathbf{\Omega}$ (Equation (B.4))—is also included for this component.

A revealing result is that, for all mechanisms, upon entering the saturated regime, the budget is dominated by Component 4 (evaluation of the production rates). In fact, its contribution in both pre-saturated and saturated regimes grows with larger mechanisms, and the effect of saturation on the budget itself (i.e., the amount that the contribution of Component 4 changes upon entering the saturated regime) is diminished when moving from mechanism Class A to Class B and then to Class C.

Therefore, it can be concluded to good confidence that the main culprit for the diminishing speedup seen in large mechanisms in Fig. B.8 is derived from Component 4. This is the direct cause of prohibitive dense cuBLAS DGEMM operations involving the exceedingly large reaction matrices (matrices of size $N_C \times N_R$) found in Equations (B.2) and (B.4). Such operations involve matrix multiplications with the net, forward, and reverse stoichiometric matrices $\boldsymbol{\nu}$, $\boldsymbol{\nu}'$ and $\boldsymbol{\nu}''$.

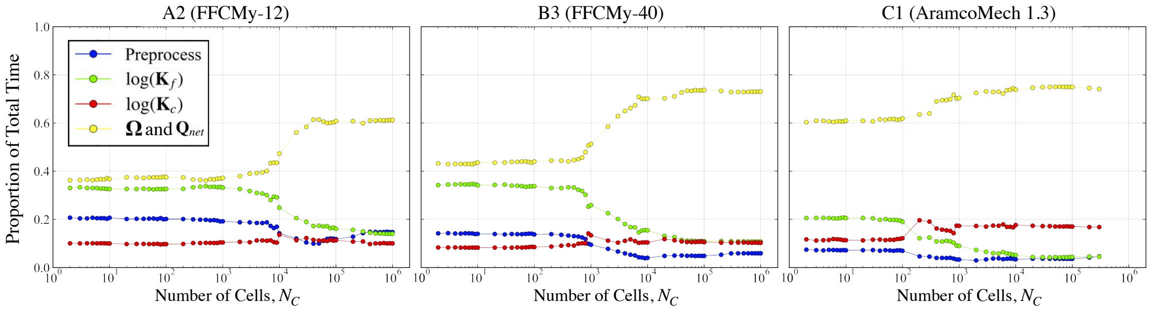


Figure B.11: Computational budget (routine time normalized by total compute time) for the preprocessing routine (Component 1, blue curve), forward rate constant routine (Component 2, green curve), equilibrium rate constant routine (Component 3, red curve), and net production rate routine (Component 4, yellow curve). The results are shown for the same three mechanisms as in Fig. B.9.

As alluded to above, with the contribution of Component 4 on the budget in mind, the diminishing speedups can be alleviated by recognizing that the common denominator in the prohibitive routines—the stoichiometric matrices—are predominantly sparse for large mechanisms due to the physical constraints of elementary reactions that serve as the building blocks. This effect can be visualized through the inherent correlation between mechanism sparsity, as measured by the ratio of the number of zero to total elements in the net stoichiometric matrix $\boldsymbol{\nu}$, and the N_C-averaged budget contribution for component 4[2]. This correlation is shown in Fig. B.12 (left). The implication is that the primary reason for the rise in Component 4 budget, and thus the main contributor to the diminishing speedup for large mechanisms observed in Fig. B.8, is the sparsity. Fig. B.12 shows that the sparsity: (a) is quite significant (roughly 60%) even for small mechanisms; and (b) converges to nearly 100% with increasing N_R.

The high sparsity for large mechanisms implies an abundance of wasted arithmetic taking place in cuBLAS-based DGEMMs, which are dense operations, used for the large reaction matrix mutiplications involving the stoichiometric matrices. Therefore, a natural step is to consider different matrix multiplication algorithms

[2] the sparsity of the net stoichiometric matrix is a conservative estimate for overall mechanism sparsity, since the forward and reverse stoichiometric matrices individually are generally more sparse

that are tailored towards sparse-dense matrix products for the larger mechanisms. This brings forward one of the most useful qualities of the matrix-oriented formulations of Sec. B.1.1—the mechanism sparsity can be integrated into the source term computation without altering the underlying matrix-based methodology. Instead, the backend used for the matrix multiplications affected by sparsity (i.e., Equations (B.2) and (B.4)) can simply be changed from cuBLAS to cuSPARSE, a GPU-based linear algebra library optimized for sparse computations also developed by NVIDIA.

The speedup achieved when moving to a sparse (driven by cuSPARSE) from a dense (driven by cuBLAS) matrix multiplication algorithm for Equation (B.2) is shown in Fig. B.12 (right)[3]. Note that the speedup shown in Fig. B.12 (right) compares two routines both executed on the GPU. Without delving into the specifics, the ultimate goal when switching to sparsity-based routines is that the savings in FLOPs should outweigh the new costs arising from algorithmic complexities and data retrieval. Based on these constraints, a general rule-of-thumb is to only use sparse algorithms when the sparsity is roughly above 95%. This quality is observed in Fig. B.12, which showcases significant matrix multiplication speedups for Mechanisms C1 (2×) and C2 (4×) in the saturated regime. Note that, when computing the sparse-to-dense speedup, the sparse compute times also include a necessary matrix transpose operation due to the fact that the cuSPARSE algorithm supports sparse-dense and not dense-sparse matrix multiplications (i.e., Equation (B.2) is transposed in order to use the cuSPARSE algorithm, and then the output is again transposed to conform to the data structures used in the authors' API—all of this is taken into account when computing the speedup in Fig. B.12).

The translation of the results in Fig. B.12 into overall GPU-to-CPU speedup analogous to the original Fig. B.8—but instead using cuSPARSE as the backend for

[3] Sparse-dense matrix multiplications are actively researched in the field of GPU computing, and there are many algorithms available. The one used here is based on the double-precision block sparse row-format matrix-multiplication (DBSRMM).

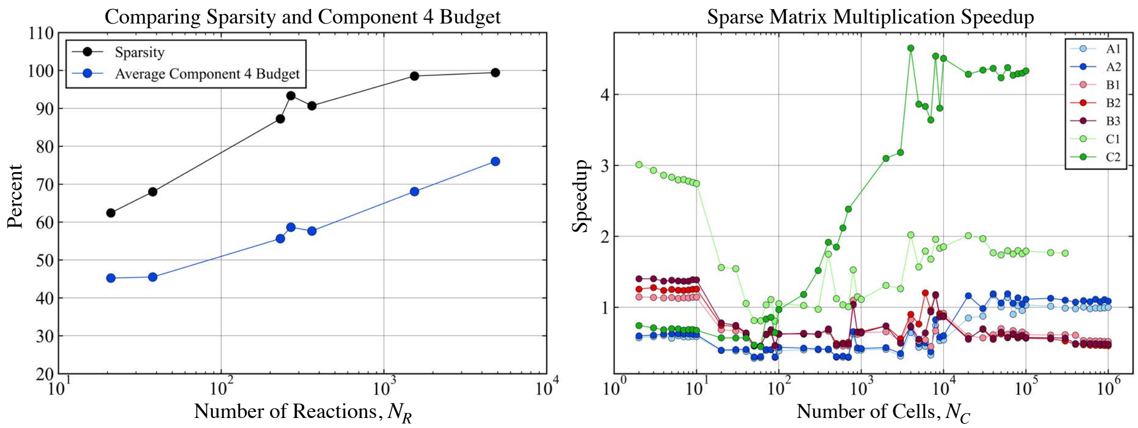


Figure B.12: (**Left**) Mechanism sparsity and average component 4 budget (the budget corresponding to the evaluation of Equations (B.2) and (B.4)) versus number of reactions N_R. (**Right**) Dense (cuBLAS) to sparse (cuSPARSE) matrix multiplication speedup (dense compute times divided by sparse compute times) for evaluating Equation (B.2) shown for all mechanisms with respect to N_C. See Table B.1 for mechanism information.

operations involving stoichiometric matrices—is shown in Fig. B.13. The results show that taking into account the mechanism sparsity alleviates the speedup deterioration seen before for the larger mechanisms of Class C. More importantly, all mechanisms regardless of size converge to similar speedup values in the saturated regime. Overall, it is encouraging that the relatively simple cuSPARSE backend modification significantly improves the GPU behavior for larger mechanisms. This warrants a more detailed analysis of the sparse algorithms themselves in relation to the stoichiometric matrix sparsity structures, since the distribution of non-zero values in the sparse matrix also affects the speedup. Such an analysis is left for future work.

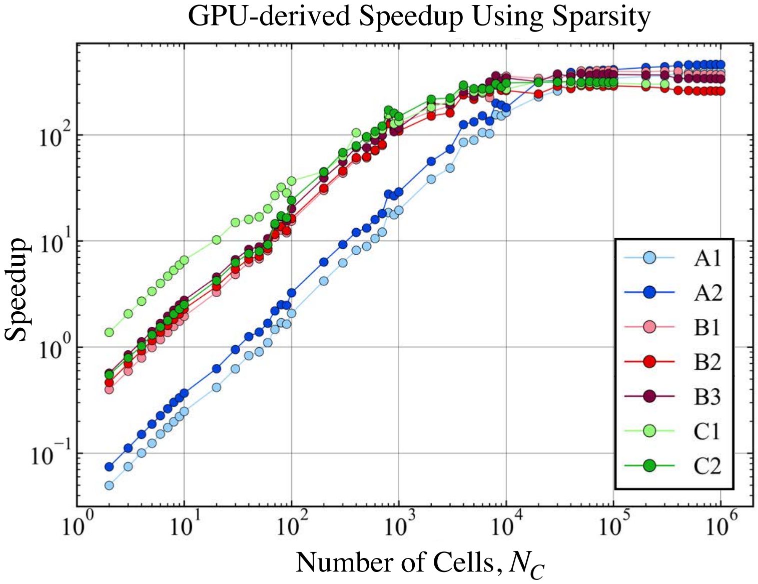


Figure B.13: Updated GPU-to-CPU speedup values when taking into account the mechanism sparsity (analogous to Fig. B.8).

APPENDIX C

GPU-Based Chemical Time Integration for Compressible Reacting Flow

This section outlines a GPU-based chemical time integration algorithm that is designed to be compatible with the matrix-based formulation for source term evaluation in Appendix B, as well as the methods for modeling chemical source terms in Chapters IV and V. In other words, the purpose of this Appendix is to show how one can deploy methods for instantaneous source term evaluations into a GPU-optimal time integration algorithm. The main idea in the time integration is to ensure that threads vectorized over the N_C cells are doing the same amount of work at all times, a nontrivial requirement in applications characterized by localized stiffness and heat release rate (i.e. shock-dominated reacting flow). Particular emphasis is placed on (a) vectorization over the number of cells, N_C, (b) treatment of GPU saturation limits, and (c) eliminating as many host-to-device data transfer penalties as possible. In the text below, some background regarding the technical challenge of stiff time integration is provided, and conventional methods of stiff chemical time integration are discussed. Then, the approach utilized in this work is presented from two perspectives: (1) a static-cell algorithm (Sec. C.3.1), which does not consider the fact that some cells in the domain are quicker to react than others, and (2) an adaptive-cell algorithm (Sec. C.3.2), which adapts to the number of chemically active cells in the domain, and allows for hybrid CPU/GPU treatment of chemical time integration to maximize speedup.

C.1 Context and Overview

Broadly speaking, there are two pathways to treat chemistry effects in a compressible reacting flow solver that handles advection and diffusion forces explicitly (see Fig. 2.5 and associated discussion). The first is a fully-coupled method-of-lines strategy, where the chemical source terms are computed instantaneously and added to the right-hand-side. This is less practical due to chemical timescale restrictions – if the chemical timescales are orders of magnitude smaller than the acoustic timescales (which is commonplace), the added computational costs and associated increase in boundary condition updates makes this approach impractical.

The second approach uses a de-coupled strategy such as operator splitting (described in Sec. 2.4.6). This requires the flow solver to call a cell-localized chemistry time integration routine (i.e reactor simulations batched over the number of cells N_C) that outputs the "reacted" thermochemical state data, which is then fed into the stable explicit solver for treated advection and diffusion contributions (see Fig. C.1). The algorithms presented here are tailored to flow solvers that invoke this type of strategy (or, in a more general sense, require some form of a batched reactor solve).

The chemistry integration step amounts to solving

$$\frac{d(\rho \mathbf{Y})_i}{dt} = \mathbf{\Omega}_i, \tag{C.1}$$

and

$$\frac{dT_i}{dt} = -\frac{1}{c_v(\mathbf{Y}_i, T_i)} \sum_{k=1}^{N_S} \frac{\epsilon_k(T_i)}{W_k} \mathbf{\Omega}_{i,k}. \tag{C.2}$$

In Eq. C.1, i denotes the cell index, $\rho \mathbf{Y} \in \mathbb{R}^{N_C \times N_S}$ is the species mass fraction matrix,

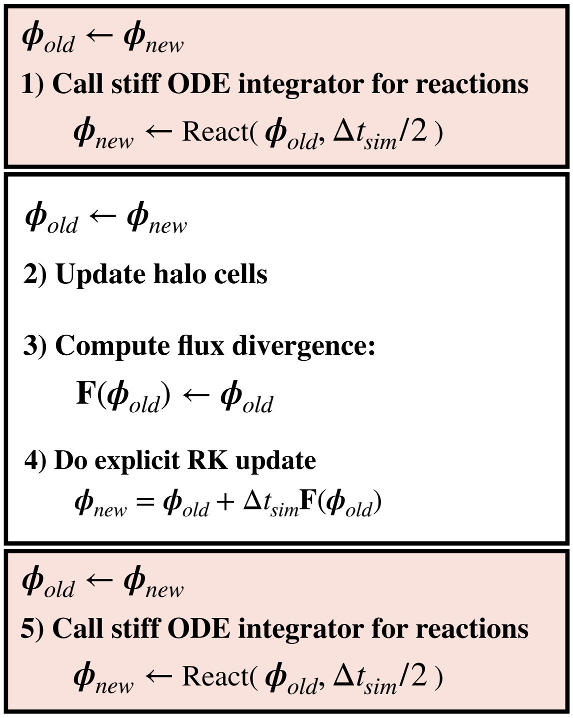

$\boldsymbol{\phi}_{old} \leftarrow \boldsymbol{\phi}_{new}$
1) Call stiff ODE integrator for reactions
$\boldsymbol{\phi}_{new} \leftarrow \text{React}(\boldsymbol{\phi}_{old}, \Delta t_{sim}/2)$
$\boldsymbol{\phi}_{old} \leftarrow \boldsymbol{\phi}_{new}$
2) Update halo cells
3) Compute flux divergence:
$\mathbf{F}(\boldsymbol{\phi}_{old}) \leftarrow \boldsymbol{\phi}_{old}$
4) Do explicit RK update
$\boldsymbol{\phi}_{new} = \boldsymbol{\phi}_{old} + \Delta t_{sim}\mathbf{F}(\boldsymbol{\phi}_{old})$
$\boldsymbol{\phi}_{old} \leftarrow \boldsymbol{\phi}_{new}$
5) Call stiff ODE integrator for reactions
$\boldsymbol{\phi}_{new} \leftarrow \text{React}(\boldsymbol{\phi}_{old}, \Delta t_{sim}/2)$

Figure C.1: Overview of reaction-advection-reaction Strang splitting algorithm. The variable $\boldsymbol{\phi}$ denotes the set of all transported state variables stored on the N_C grid points. The reaction step (red box) is local in physical space, but non-local in thermochemical phase space. The advection/diffusion step is non-local in physical space, but local in thermochemical in phase space.

and $\boldsymbol{\Omega} \in \mathbb{R}^{N_C \times N_S}$ denotes the species net production rates (units of $\frac{kg}{m^3 s}$) derived from the Arrhenius-based reaction network. In Eq. C.2, c_v denotes the mass-based specific heat at constant volume for the mixture, ϵ_k is the molar internal energy for species k (derived from NASA polynomials), W_k is the species molecular weight, and $\boldsymbol{\Omega}_{i,k}$ is the production rate (chemical source term) for species k.

Equations C.1 and C.2 define a constant-volume reactor – in other words, the cell density ρ_i is held constant throughout the chemical time integration. It is possible to also treat constant-pressure reactors instead. However, it can be argued that a constant-volume approach is more consistent with the governing equations in the operator splitting framework, since the chemical integration phase should not be updating the mixture density. The consequence is that the underlying assumption imposed here is (a) fixed total internal energy (sensible + chemical), (b) variable pressure (by means of the EoS), and (c) variable total enthalpy throughout the chemistry integration.

This Appendix operates under the context that Eq. C.1 is solved using an adaptive-timestep explicit algorithm, and Eq. C.2 is replaced with an implicit Newton-Raphson iterative solve based on the constant internal energy criterion [30]. In other words, given an input species mass fraction vector and mixture internal energy, the functional form of internal energy – described via NASA polynomials as an explicit functions of temperature and mass fraction – is inverted to recover the temperature to a negligibly small tolerance.

It can be argued that an explicit treatment of Eq. C.1 does not directly address the issue related to prohibitive chemical timescales. Alternatives include using a fully implicit time integrator based on backward differentiation formulas [63], or using a manifold-based model to decouple the fast and slow scales [101]. Although this argument is valid, the computational advantage provided by the above methods within the GPU scope is unclear – further GPU-targeted analysis of the tradeoff between large timestep allowance (as provided by CVODE and CSP) and higher, potentially prohibitive algorithmic costs introduced by these methods (memory limitations, start-up costs, Jacobian evaluation, eigendecompositions, etc.) must be performed with openly available GPU libraries that implement the above methods (i.e. TINES [149]). These issues may be amplified when considering the fact that in the shock-dominated compressible reacting flow applications described in Sec. 1.2 (e.g. RDEs and scramjets), one often observes space-localized stiffness, where only a small fraction of computational cells carry prohibitive chemical timescales. As such, the time-adaptive explicit algorithm for Eq. C.1 is retained here – as described below, explicit treatment brings forward the ability to create highly vectorized offloading strategies for the chemical time integration whose computational efficiency offsets potential timestep restrictions.

C.2 Conventional CPU-based Algorithm

A conventional CPU-based time integration algorithm is shown in Alg. 3. For brevity, this algorithm displays an explicit Euler integration step – extensions to higher-order Runge-Kutta schemes are straightforward. The inputs to the routine are density, internal energy (sensible + chemical), temperature, and species mass fraction (ρ, e, T, and $\mathbf{Y}$ respectively) at simulation time t. The outputs are both temperature T and species mass fraction $\mathbf{Y}$ at the time $t + \Delta t_{sim}$ (note that within the context of the second order Strang splitting scheme of Fig. C.1, the target time can be modified to $t + \Delta t_{sim}/2$ without loss of generality). The local timestep, denoted Δt_{local}, is the cell-dependent timestep used to carry out the integration.

In a general sense, within the for loop over all cells, the user calls a per-cell chemistry time integrator which is represented here as a while loop. The number of while loop iterations per-cell denotes implicitly a degree of stiffness associated with the initial thermochemical state of the cell. Step (e) ensures that no overshoot occurs, and step (f) updates the local time step using a criterion based on the source term magnitude: the local time step is set such that a given species mass fraction does not change by more than some percentage (typically $1 - 5\%$) of its current value. Candidate timesteps are obtained for all species, followed by a reduction over the number of species to determine the final timestep (the minimum is used for all species). Note that, although commonly used, prescribing a timestep in this fashion estimates the chemical timescales with a diagonal Jacobian approximation (cross-species sensitivities are not accounted for). Despite this, timestep estimations of this type are known to work well in combustion and astrodynamical applications, and reduce the need to decompose chemical Jacobians. Given Alg. 3, a user typically has two offloading strategies.

C.2.1 Cell-local Offloading Strategy

We recognize that the computational bottleneck comes from the source term evaluation in Step (d). As such, a common approach is to offload this function *within the while-loop* to the GPU, and keep all remaining elements on the CPU. Here, the scope of the GPU-based evaluation of the source terms occurs on a per-cell basis, which means the call to the GPU library must occur for each cell individually. Although this is a fairly common and non-intrusive approach in terms of code re-structuring requirements, it is not ideal for a few reasons. The first is that the GPU evaluation is not vectorized over the number of cells N_C, which means the user cannot rely on N_C to control GPU saturation levels. In situations where $N_C \gg N_S$ (the case in practically all compressible reacting flow simulations), the GPU resources will be left underutilized. Alternatively, the user must rely on very detailed mechanisms to enable GPU saturation (i.e. N_S on the order of 1000 or higher). Although this may provide saturation, the host-to-device (and vice versa) data transfer penalties will negate any computational advantage provided by the source term offloading, since the number of data transfers in this strategy scales linearly with both the number of cells and the number of while-loop iterations per cell.

C.2.2 Single-Kernel Offloading Strategy

The entire for loop over cells is treated as a unified GPU kernel. Here, a single GPU thread executes the entire while loop contained in Alg. 3. Although this solves the issues related to data transfer penalties, and also addresses the issue of cell-based vectorization, the tradeoff is that the scope of each thread is too large. In other words, treating Alg. 3 with a single large kernel will lead to thread concurrency issues which reduces speedup [16].

Due to these limitations, Alg. 3 must be redesigned to maximize thread concurrency and minimize data transfers. This is described in the following section.

Algorithm 3 Conventional explicit chemistry time integration.

```
Data: $\rho(t) \in \mathbb{R}^{N_C \times 1}$, $e(t) \in \mathbb{R}^{N_C \times 1}$, $T(t) \in \mathbb{R}^{N_C \times 1}$, $\mathbf{Y}(t) \in \mathbb{R}^{N_C \times N_S}$
Result: $\mathbf{Y}(t + \Delta t_{sim})$, $T(t + \Delta t_{sim})$
for $i$ in range($N_C$) do
    a) Initialize local time: $t_{local,i} = 0$
    while $t_{local,i} < \Delta t_{sim}$ do
        b) Correct temperature: $T_i \leftarrow NewtonRaphson(T_i, e_i, \mathbf{Y}_i)$
        c) Update pressure: $p_i \leftarrow \rho_i R T_i / W_i$
        d) Get species source terms: $\mathbf{\Omega}_i \leftarrow netProdRates(p_i, T_i, \mathbf{Y}_i)$
        e) Initialize integration timestep: $\Delta t_{local,i} \leftarrow \Delta t_{sim} - t_{local,i}$
        f) Adjust integration timestep: $\Delta t_{local,i} \leftarrow adjustDt(\mathbf{\Omega}_i, \rho_i, \mathbf{Y}_i)$
        g) Advance mass fractions: $\rho \mathbf{Y}_i \leftarrow \rho \mathbf{Y}_i + \Delta t_{local,i} \mathbf{\Omega}_i$
        h) Update local time: $t_{local,i} \leftarrow t_{local,i} + \Delta t_{local,i}$
    end
    i) Correct temperature (see (b)).
end
```

C.3 Vectorized GPU-based Algorithm

In this section, a modified version of the conventional algorithm discussed in Sec. C.2 is presented. Although the underlying structure of the time integration is kept the same, some key modifications are required to enable full vectorization over N_C and to ensure GPU-optimality. The primary issue to address is the fact that some cells are quicker to react than others – this quality introduces many viable algorithm variations. Two such variations of the GPU-based algorithm are provided here, both of which are used in the GPU solver UMReactingFlow as described in Ref. [30] depending on the application: (1) a *static-cell algorithm*, which does fixes the number of cells offloaded to the GPU regardless of reaction completion (N_C does not change in time), and (2) an *adaptive algorithm*, which both adapts to the number of chemically active cells in the domain (i.e. N_C changes in time within the chemistry integration phase), and also allows for hybrid CPU/GPU treatment of chemical time integration to maximize speedup.

C.3.1 Static GPU Algorithm

The goal of this algorithm is to maximize the thread concurrency and expose as much parallelism as possible during the chemistry time integration phase. To properly vectorize Alg. 3 over the number of cells N_C, an indicator function $\mathcal{F}$ is introduced such that

$$\mathcal{F}_i = \begin{cases} 0, & \text{if } t_{local,i} = \Delta t_{sim}, \\ 1, & \text{if } t_{local,i} < \Delta t_{sim}, \end{cases} \tag{C.3}$$

where $i = 1, \ldots, N_C$. The indicator $\mathcal{F}$ encodes whether or not a cell has finished reacting. Since $\mathcal{F} \in \mathbb{Z}^{N_C}$, we also have

$$N_C^R = \sum_i \mathcal{F}_i, \quad N_C^{NR} = N_C - N_C^R, \tag{C.4}$$

where N_C^R denotes the number of cells that are still undergoing chemical reaction, and N_C^{NR} are the cells that have finished reacting.

The algorithm is provided in Alg. 4. **Note that, although not shown in Alg. 4, every step within the while loop is a batched operation over all N_C cells, regardless of the value of N_C^R.** To expose as much vectorization over N_C as possible, this algorithm essentially moves the inner while loop from Alg. 3 outside of the for loop instances. This is possible through the reaction progress indicator function, $\mathcal{F}$, and the associated metric for evaluating the number of reacting cells (Eq. C.4). The main idea is to terminate the while loop only when the number of reacting cells, N_C^R, reaches zero.

Every step within the while loop in Alg. 4, which is a for loop instance, is interpreted as a kernel to be executed on the GPU. The reason this is called a "static-cell" algorithm is because, regardless of the value N_C^R, all kernels are threaded over the total number of cells N_C. Although this approach leads to wasted arithmetic operations, it is acceptable for scenarios in which the GPU is in an undersaturated regime where the kernel evaluation times are not functions of N_C.

The crucial kernels in Alg. 4 are described below:

- **Alg. 4(g)**: GPU-based evaluation of source terms is performed with the matrix-based formulation described in Ref. [16]. The source term evaluation is treated as a batched operation over the number of cells N_C – optimal acceleration is enabled by means of exposing general matrix multiplications (GEMMs) throughout the computation (e.g. for reaction rate and rate constant evaluation). These GEMMs are treated with cuBLAS/cuSPARSE linear algebra libraries.

- **Alg. 4(h)**: This step utilizes multiplication by indicator $\mathcal{F}$ to force species source terms for completed cells to zero. This ensures completed cells are not advanced in the explicit update step (k).

- **Alg. 4(k)**: The mass fractions are advanced in an explicit update step. This explicit advance can be treated as a batched double-precision ”AX-Plus-Y” (DAXPY) operation using a linear algebra library like cuBLAS, and contributes very little to overall cost.

- **Alg. 4(n)**: The while loop argument N_C^R – the number of chemically reacting cells – is updated via summation of the indicator variable $\mathcal{F}$. This is a GPU-based reduction of an integer array; as such, its cost is minimal. Upon reduction, the new N_C^R must be sent back to the CPU to proceed with the while loop (this constitutes a single integer data transfer operation).

C.3.2 Adaptive GPU Algorithm

For situations in which the GPU is heavily saturated and there are a small number of stiff cells that require significantly higher number of while loop iterations than the rest, the static-cell approach discussed in Sec. C.3.1 is less ideal. Instead, the algorithm must be modified to ensure that the for loops in Alg. 4 operate over N_C^R instead

Algorithm 4 Static-Cell GPU Time Integration. Although not shown, each step within while loop is a for loops over all N_C cells.

```
Data: $\rho(t) \in \mathbb{R}^{N_C \times 1}$, $e(t) \in \mathbb{R}^{N_C \times 1}$, $T(t) \in \mathbb{R}^{N_C \times 1}$, $\mathbf{Y}(t) \in \mathbb{R}^{N_C \times N_S}$
Result: $\mathbf{Y}(t + \Delta t_{sim})$, $T(t + \Delta t_{sim})$
Begin Initialization:
  a) Perform host-to-device data transfers
  b) Set local time: $\forall i, t_{local,i} \leftarrow 0$
  c) Set indicator (Eq. C.3): $\forall i, \mathcal{F}_i \leftarrow 1$
  d) Set $N_C^R$: $N_C^R \leftarrow N_C$
Begin Time Integration:
  while $N_C^R > 0$ do
      e) Correct temperature: $T_i \leftarrow NewtonRaphson(T_i, e_i, \mathbf{Y}_i)$
      f) Update pressure:$p_i \leftarrow \rho_i R T_i / W_i$
      g) Get species source terms (see Appendix B): $\mathbf{\Omega}_i \leftarrow netProdRates(p_i, T_i, \mathbf{Y}_i)$
      h) Zero source terms if complete: $\mathbf{\Omega}_i \leftarrow \mathbf{\Omega}_i \mathcal{F}_i$
      i) Initialize integration timestep: $\Delta t_{local,i} \leftarrow \Delta t_{sim} - t_{local,i}$
      j) Adjust integration timestep: $\Delta t_{local,i} \leftarrow adjustDt(\mathbf{\Omega}_i, \rho_i, \mathbf{Y}_i)$
      k) Advance mass fractions: $\rho \mathbf{Y}_i \leftarrow \rho \mathbf{Y}_i + \Delta t_{local} \mathbf{\Omega}_i$
      l) Update local time: $t_{local} \leftarrow t_{local} + \Delta t_{local}$
      m) Update indicator function: $\mathcal{F}_i \leftarrow$ Eq. C.3
      n) Update $N_C^R$ (reduction, Eq. C.4): $N_C^R \leftarrow reduceSum(\mathcal{F})$
end
o) Correct temperature: $T_i \leftarrow NewtonRaphson(T_i, e_i, \mathbf{Y}_i)$
```

Algorithm 5 Adaptive-Cell GPU/CPU Time Integration

```
Data: $\rho^*(t) \in \mathbb{R}^{N_C^A \times 1}$, $e^*(t) \in \mathbb{R}^{N_C^A \times 1}$, $T^*(t) \in \mathbb{R}^{N_C^A \times 1}$, $T(t) \in \mathbb{R}^{N_C \times 1}$, $\mathbf{Y}^*(t) \in$
      $\mathbb{R}^{N_C^A \times N_S}$, $\mathbf{Y}(t) \in \mathbb{R}^{N_C \times N_S}$, $N_{sat}$, $N_{CPU}$, $UF$
Result: $\mathbf{Y}(t + \Delta t_{sim})$, $T(t + \Delta t_{sim})$
Begin Initialization:
  a) Perform host-to-device data transfers
  b) Set local time: $\forall i, t_{local,i} \leftarrow 0$
  c) Set indicator (Eq. C.3): $\forall i, \mathcal{F}_i \leftarrow 1$
  d) Set $N_C^R$: $N_C^R \leftarrow N_C$
  e) Set $N_C^A$: $N_C^A \leftarrow N_C$
  f) Set CPU flag: useCpu ← False
  g) Set loop counter: $counter \leftarrow 0$

Begin Time Integration:
  while $N_C^R > 0$ do
    h) Switch to CPU if needed:
      if $N_C^A \leq N_{CPU}$ then
        useCpu ← True
          Exit while loop.
    end
    i) Do steps (e)-(n) from Alg. 4 using re-sized inputs $\rho^*$, $e^*$, $T^*$, $\mathbf{Y}^*$ (kernels
      executed over $N_C^A$ instead of $N_C$).
      j) Accumulate loop counter: $counter \leftarrow counter + 1$.
      k) Re-arrange state data (Fig. C.2(b)): $N_C^A$: $N_C^A \leftarrow N_C^R$
end
l) React remaining cells on CPU if needed:
  if $useCPU = True$ then
    l-1) Device-to-host data transfer: send input state data from CPU to GPU.
      l-2) Execute Alg. 3 on CPU for remaining $N_C^A$ cells.
      l-3) Host-to-device data transfer: send reacted state data from GPU to CPU.
end
m) Perform final copy (see k-1-1).
  n) Correct temperature (see Alg. 4(o)).
```

of N_C. This modification, presented as the adaptive-cell algorithm in this section, is nontrivial because N_C^R changes in time (some cells finish reacting before others); this quality requires on-the-fly re-arrangement of data on the GPU (see Fig. C.2(b)) in order to satisfy the memory-contiguous requirements of kernel inputs. This data rearrangement step is the key difference from the static algorithm of Sec. C.3.1 – in the end, the savings obtained from adaptively adjusting the number of reacting cells should overcome the additional bandwidth-limited cost associated with re-arranging the data on the GPU. The adaptive algorithm also takes into account that the CPU executes time integration faster than the GPU for a very small number of cells (e.g. 5-10); as such, when the number of reacting cells is sufficiently small, the time integration is switched from the GPU to the CPU, resulting in a hybrid algorithm.

The adaptive algorithm is summarized in Alg. 5. The main difference from the static algorithm (Alg. 4) is the introduction of N_C^A – the number of active cells – which replaces N_C during the time integration step (instead of threading over N_C during the explicit update, the adaptive algorithm threads over N_C^A which is smaller than N_C for most of the time integration phase). The number of active cells N_C^A is re-synchronized to N_C^R (the number of cells that have finished reacting) in a data-rearrangement step every UF while-loop iterations, where UF is the so-called *update frequency* to be supplied by the user as input. For example, $UF = 10$ ensures that every 10 while loop iterations, a check is made to see if N_C^A can be reduced further. As described further below, if this check (based on the GPU saturation point) is satisfied, a data rearrangement step takes place (see Fig. C.2(b)) and N_C^A is reduced to N_C^R. Some key steps in Alg. 5 are described in more detail below:

- **Input Parameters:** The algorithm requires three user supplied inputs: N_{sat}, N_{CPU}, and UF. N_{sat} is the GPU saturation point with respect to number of cells (here N_C^A) for the vectorized chemical source term evaluation, which is the most computationally intensive step in the time integration procedure. The

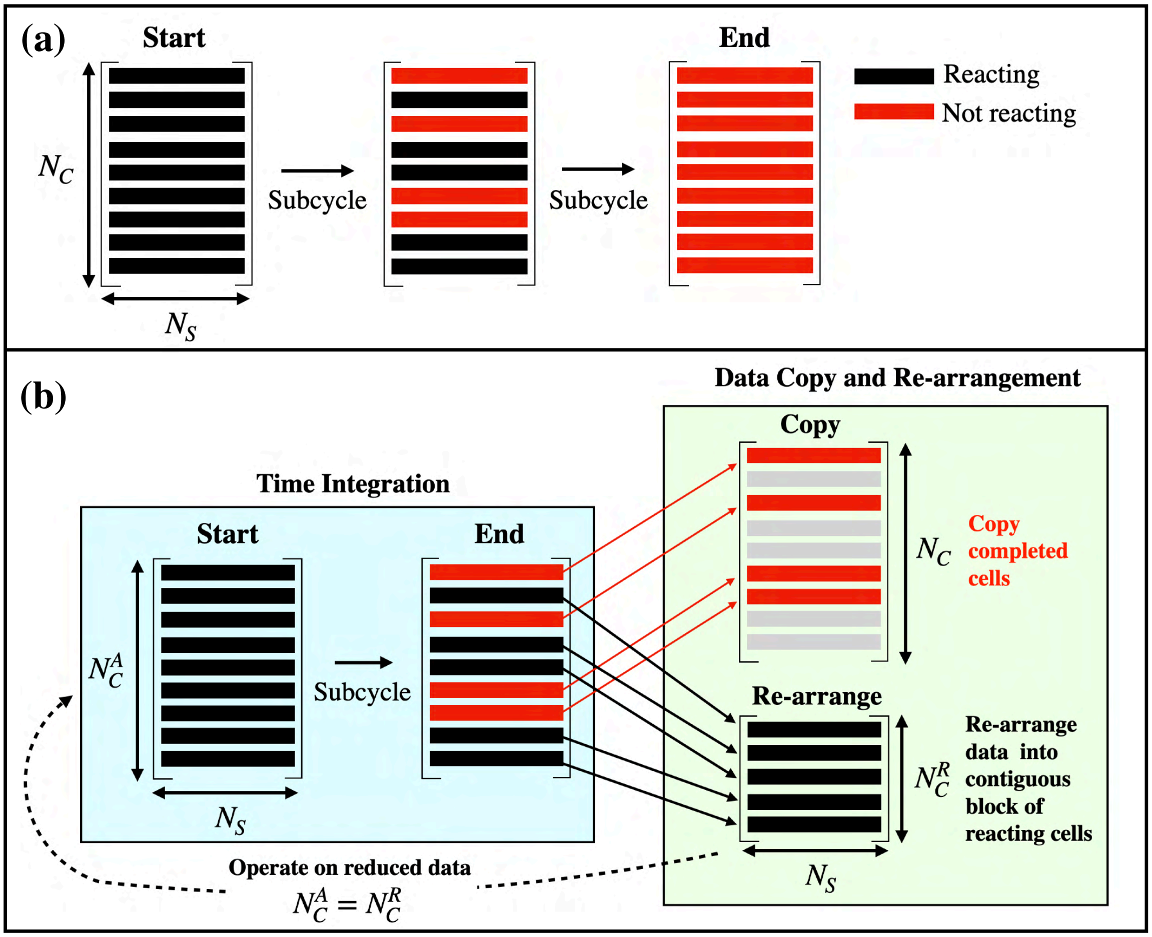


Figure C.2: **(a)** Schematic of the static-cell time integration algorithm: as subcycles (while loop iterations) proceed, some cells finish reacting before others. The static-cell algorithm fixes N_C and assigns zero source terms to completed cells. **(b)** Schematic of the adaptive-cell algorithm. There are two stages: the time-integration stage (blue box) and the data re-arrangement stage (green box). The time-integration stage is the same as **(a)**, but it operates on the quantity N_C^A instead of N_C; N_C^A is reduced on-the-fly in the re-arrangement stage.

saturation point is the cell value below which the time-to-solution for source term evaluation is independent of N_C^A. N_{CPU} is the cell value at which the CPU-based source term evaluation is faster than the GPU-based evaluation. See Sec. C.4 for further clarification on saturation and speedup trends. Lastly, UF is the update frequency; it sets the frequency (in terms of every UF while loop iterations) at which the algorithm checks if a reduction to N_C^A is necessary. N_{sat} and N_{CPU} will depend on the chemical mechanism complexity in terms of N_S (number of species) and N_R (number of reactions). UF is purely user-dependent: a high UF will ensure that minimal data rearrangement steps are executed at the cost of potential wasted operations, and a small UF (say $UF = 1$) ensures that data rearrangement steps are executed as often as possible, minimizing wasted operations but maximizing frequency of bandwidth-limited data rearrangement operations. Table C.1 shows how to recover the static-cell (Alg. 4) and conventional CPU-based (Alg. 3) routines directly from Alg. 5 by means of appropriate parameter settings.

- **Alg. 5(h)**: If the number of active cells N_C^A drops below the value N_{CPU}, the conventional CPU-based routine in Alg. 3 is used to treat the remaining cells.

- **Alg. 5(i)**: The same explicit update steps described in the static algorithm (Alg. 4) are used here – the difference is that the for loops in Alg. 4(e)-(n) operate over N_C^A instead of N_C.

- **Alg. 5(k)**: Every UF while loop iterations, a set of update conditions based on saturation trends is checked to determine if a data rearrangement on the GPU is necessary. These checks are as follows:

  1. $N_C^{NR} < N_C^A$: The update should only occur when the number of non-reacting (or complete) cells is lower than the current number of active cells.

Description	Parameters
1) Adaptive, Hybrid GPU/CPU	$1 < N_{CPU} < N_{sat}$
2) Adaptive, GPU only	$N_{CPU} = -1$, $N_{sat} \geq 1$
3) Static, GPU only	$N_{CPU} = -1$, $N_{sat} = \text{Inf}$
4) Conventional, CPU only	$N_{CPU} = \text{Inf}$

Table C.1: Implications of various parameter configurations in Alg. 5. Option 3 recovers the static algorithm (Alg. 4) and Option 4 recovers the conventional CPU algorithm (Alg. 3). If the GPU is heavily saturated, the ideal configuration is adaptive (Options 1 and 2). If GPUs are undersaturated, performance difference between (1/2) and (3) is expected to be minimal.

2. $N_C^A > N_{sat}$: The update should only occur when the current number of active cells is above the GPU saturation point for source term evaluation. This is described in further detail in Sec. C.4.

If the conditions described above are satisfied, the following steps are executed to update the number of active cells N_C^A. First, the thermochemical state data corresponding to the completed cells (of N_C^{NR} amount) are transferred to a set of "final" matrices (of size N_C) that stores the end result. Second, the remaining N_C^R cells (still reacting) are re-formatted to occupy a contiguous memory block. Third, the number of active cells N_C^A is reduced to N_C^R, which concludes the while loop iteration. This process is illustrated in Fig. C.2(b).

C.4 Saturation and Speedup Trends

A discussion on GPU saturation and speedup trends is described below to properly motivate the data re-arrangement routine in the adaptive algorithm. This is intended to outline how one can choose the N_{CPU} (cell value at which the CPU takes over the time integration from the GPU) and N_{sat} (cell value at which the GPU is fully saturated) input parameters required by the adaptive algorithm.

Figure. C.3 illustrates the effect of GPU saturation on both absolute compute time and speedup over CPU for a given kernel. The characteristic kernel here is the evaluation of the chemical source terms, which is the most compute-intensive step in the time integration algorithm. As such, the upper plot in Fig. C.3 corresponds to time-to-solution for source term evaluation, and the bottom plot corresponds to speedup over an unvectorized CPU-based evaluation of the source terms (e.g. from Cantera). Note that Fig. C.3 intends only to illustrate the saturation phenomenon schematically; plots derived from a true profiling study can be found in Appendix B.

The plots in Fig. C.3 can be split into undersaturated and saturated regimes on the basis of cells-per-GPU (N_C). The undersaturated regime is characterized by no change in GPU compute time with respect to N_C, whereas the saturated regime is characterized by a linear increase in absolute compute time with N_C. The saturation point indicates the N_C at which a transition between undersaturation and saturation occurs on the GPU for the kernel in question.

The adaptive GPU time integration algorithm is motivated directly by the trends shown in Fig. C.3. As chemistry time integraton proceeds, the number of reacting cells N_C^R begins to drop. This can be interpreted as a right-to-left traversal in Fig. C.3. With this in mind, the saturation and speedup trends necessitate the following three restrictions on adaptive cell updates on the GPU:

1. Update the number of cells (i.e. effectively change N_C to N_C^R) when in the saturated regime, as this will give a proportional decrease in compute time. This is the region between points A and B in Fig. C.3.

2. Do not update the number of cells when in the undersaturated regime, as this gives no additional reduction in compute time. This is the region between points B and C in Fig. C.3.

3. Offload back to the CPU when the number of reacting cells drops to the point

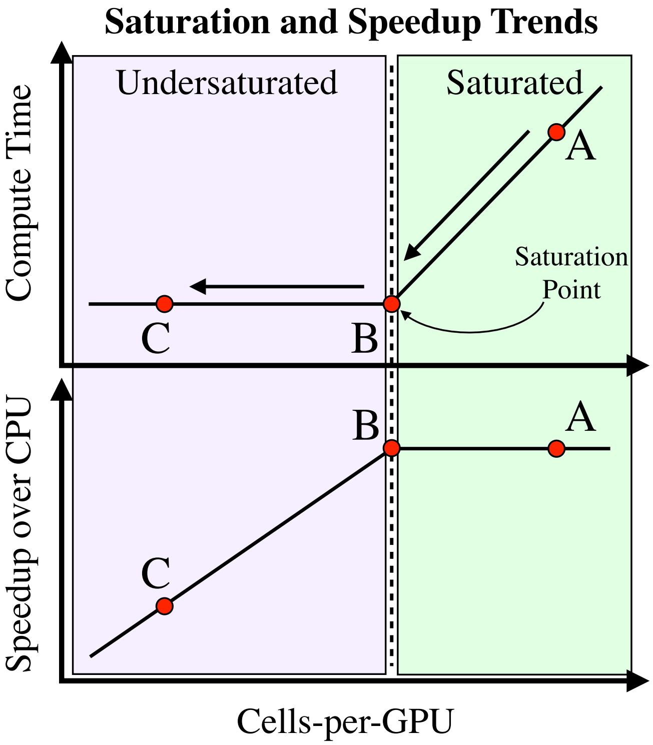


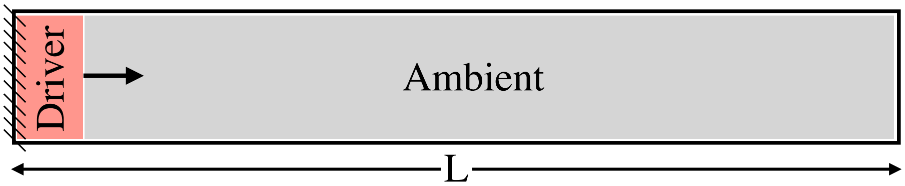
Figure C.3: Illustration of saturation effect (top) and speedup trends over CPU (bottom) versus N_C (cells-per-GPU) for a key kernel, here taken to be the chemical source term evaluation. During chemical time integration, the number of chemically active cells drops, as shown by the arrows. **Point A** is in the saturated regime. **Point B** is the saturation point. **Point C** is the N_C value at which a CPU evaluation is faster than GPU. The locations of point B and C depend on the GPU architecture and the chemical mechanism. Note that these diagrams are exaggerations of true behavior – see Ref. [16] for true profiles.

at which kernel evaluation on the CPU (host) is faster than the GPU (device). This is the region below point C in Fig. C.3. Point C occurs at very small cell numbers, typically around 5 or 10 depending on the chemistry mechanism. However, switching back to the CPU for low cell numbers can result in significant computational savings when treating highly localized chemical stiffness where a very small number of cells are responsible for the small chemical timescales. This occurs frequently in advection-dominated compressible reacting flow (e.g. detonations).

C.5 Performance Comparison

This section compares performance between the static (Alg. 4) and adaptive (Alg. 5) GPU chemistry time integration algorithms in the context of shock-dominated compressible reacting flow. The objective here is to demonstrate the advantage provided by the adaptive algorithm for problems containing highly localized heat release and stiffness, which is characteristic of the application cases considered in this dissertation as outlined in Chapter I, Sec. 1.2. As such, a targeted study based on weak scaling of 1-d detonation simulations is used to isolate the impact of localized stiffness on the performance of the time integration algorithms with detailed chemistry. All discussion hereafter applies within the context of a single MPI rank offloading the chemistry time integration task to an associated GPU in a domain-decomposition scope (e.g. the number of cells N_C represents the per-GPU amount). All simulations were performed using NVIDIA V100 GPUs.

The domain configuration is shown in Fig. C.4. Following the method of Ref. [364], detonations are initiated by means of imposing a high-energy driver gas in a small region on the leftmost side of the domain. As shown in Fig. C.4, three different domain lengths are tested at different orders of magnitude with proportional increase in the number of cells N_C. This fixes the resolution in all cases to 50 micron. As mentioned above, the idea here is to demonstrate the advantage of the adaptive algorithm by isolating purely the effect of localized stiffness. In other words, since resolution is fixed, the increase in number of cells-per-GPU (N_C) does not change the physics of the problems between cases; instead, as N_C increases, the number of chemically active cells in proportion to chemically inactive cells is expected to drop, which in turn favors the need to adopt the adaptive-cell algorithm. As such, the goal is to quantify the advantage provided by the adaptive-cell algorithm on the GPU with direct comparison to the static-cell counterpart using N_C as an effective knob for the ratio of chemically active to inactive cells via the weak scaling of 1-d detonations.



Case	N_C	L [m]	Δx [micron]
1	6,000	0.30	50
2	60,000	3.00	50
3	600,000	30.0	50

Figure C.4: 1d detonation configuration, where N_C represents the number of cells-per-GPU. Each case maintains the same resolution. Length of driver region is fixed at 3mm.

Fuel	**Oxidizer**	**Reference**	$\mathbf{N_S}$	$\mathbf{N_R}$	**Driver Gas**	$\mathbf{P_D}$ [atm]	$\mathbf{T_D}$ [K]
H2	Air	[212]	9	21	Nitrogen	100	2500
CH4	O2	[359]	12	38	CJ condition	100	2500
RP2-2	O2	[361]	38	192	CJ condition	50	3900

Table C.2: List of mechanisms used along with driver gas properties for 1d detonation configuration (see Fig. C.4). Ambient gas properties were set to equivalance ratio of unity with $T = 300$K and $P = 1$atm. The last three columns provide input parameter settings needed by the adaptive algorithm (see Alg. 5 and Sec. C.3.2). In all cases, $N_{sat} = 1000$, $N_{CPU} = 100$, and $UF = 10$.

Additionally, for each domain case in Fig. C.4 (each N_C), three chemical mechanisms are compared. These mechanisms are provided in Tab. C.2 in order of increasing complexity.

Chemical Stiffness Trends: Figure C.5 shows how both the simulation timesteps and the chemical timescales change during the time evolution of a sustained detonation for the three mechanisms listed in Tab. C.2. For each point in time, the simulation timestep is extracted from the CFL condition of 0.2 using a standard acoustic wavespeed estimate. Each cell provides a candidate timestep, and the minimum over the entire domain is ultimately chosen (shown in blue in Fig. C.5). The chemi-

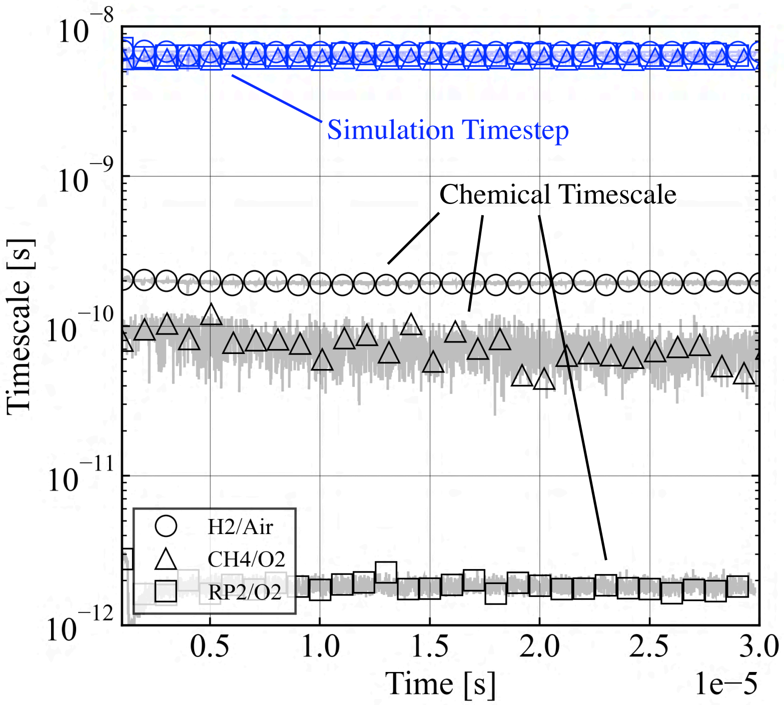


Figure C.5: Time evolution of simulation time step (blue) and chemical timescales (black) during steady detonation propagation for the three mechanisms defined in Tab. C.2. Reproduced from Fig. 1.7 for ease of readability.

cal timescales (black in Fig. C.5) are approximations; they are obtained by dividing the simulation timestep at the corresponding time instant by the number of chemistry subcycles (while loop iterations in the chemistry time integration algorithm), providing an interpretable representation of chemical stiffness within the context of sustained detonation. Fig. C.5 shows how all three mechanisms observe simulation timesteps at the same order of magnitude (roughly 6 nanoseconds), whereas the chemical timescale estimates are noticeably different. This implies that, for the purposes of this study, the selection of chemical mechanism provides a pathway for profiling the effect of stiffness on the cell-adaptive GPU time integration algorithm. The stiffness ratio – which is the ratio between simulation timestep and chemical timescale – is on the order of 10^2 for both H2/Air and CH4/O2 mechanisms (less stiff), and is roughly 10^4 for the RP2 mechanism (most stiff), providing good baselines for assessment of stiffness variation.

Speedup over Static Algorithm: Figure C.6(a) shows average speedup derived from the adaptive GPU-based chemistry time integration algorithm (Alg. 5) over the static GPU-based algorithm (Alg. 4) during the detonation simulation for increasing cells-per-GPU. The adaptive algorithm parameters for each mechanism (N_{sat}, N_{CPU}, and the update frequency UF) are provided in Tab. C.2. The speedup shown in Fig. C.6(a) comes from evaluating average times-to-solution for chemical time integration during each simulation time step. As mentioned in the discussion surrounding Fig. C.4, because the increase in cells-per-GPU N_C corresponds directly to an increase in localized stiffness (i.e. high N_C in this problem configuration drops the ratio of the number of chemically active to inactive cells), the speedup provided by the adaptive algorithm increases at constant rate with cell count for all mechanism tested. Further, the adaptive-cell algorithm provides greater speedup over the static-cell counterpart for mechanisms with higher complexity (higher values of N_C and N_R), as illustrated by the vertical shift of each curve in Fig. C.6(a). Note that for the less complex mechanisms, speedup over the static-cell algorithm is minimal when cell counts are low – for example, the adaptive-cell algorithm applied on the H2/Air and CH4/O2 mechanisms do not see signifant computational gain at low N_C. This comes from the saturation limit – the adaptive algorithm is ideal in situations where the baseline N_C (cells-per-GPU) is above the GPU saturation point. Since the saturation point decreases in proportion to the number of reactions in the chemical mechanism, the computational advantage provided by the adaptive-cell algorithm increases significantly with chemical mechanism complexity.

It should be noted that the slope in Fig. C.6(a) (the rate of increase in speedup with respect to to cells-per-GPU) increases with chemical mechanism stiffness. This effect of stiffness on speedup derived from the adaptive-cell algorithm is shown in Fig. C.6(b), which illustrates the correlation between time-to-solution for chemical time integration (x-axis) and the number of subcycle iterations (while-loop iterations, y-axis). A high subcycle iteration implies increased chemical stiffness. Figure. C.6(b) shows how the increase in chemical stiffness indicates an increase in speedup provided by the adaptive algorithm. Ultimately, the adaptive algorithm ensures that detailed, highly stiff chemical mechanisms become significantly more practical on the GPU in compressible reacting flow simulations.

Note that due to the application scope on flow solvers designed to treat advection-dominated compressible reacting flow (which is the scope of this dissertation), this performance analysis was designed to emphasize the computational advantage provided by the adaptive-cell algorithm over the static-cell counterpart in scenarios with chemical heat release highly localized in space. Similar analysis of the adaptive-cell chemical time integration algorithm for canonical problems with evenly distributed chemical reactions (e.g. Taylor-Green vortex or homogeneous isotropic turbulence) is left for future work. Preliminary analysis shows that similar trends seen in Fig. C.6 are also observed in these situations, but with lower peak speedups – for problems with spatially distributed chemical reactions dominated by turbulence, speedup over the static-cell algorithms for complex mechanisms of moderate stiffness peaks at roughly 10x, with typical values ranging between 5-7x.

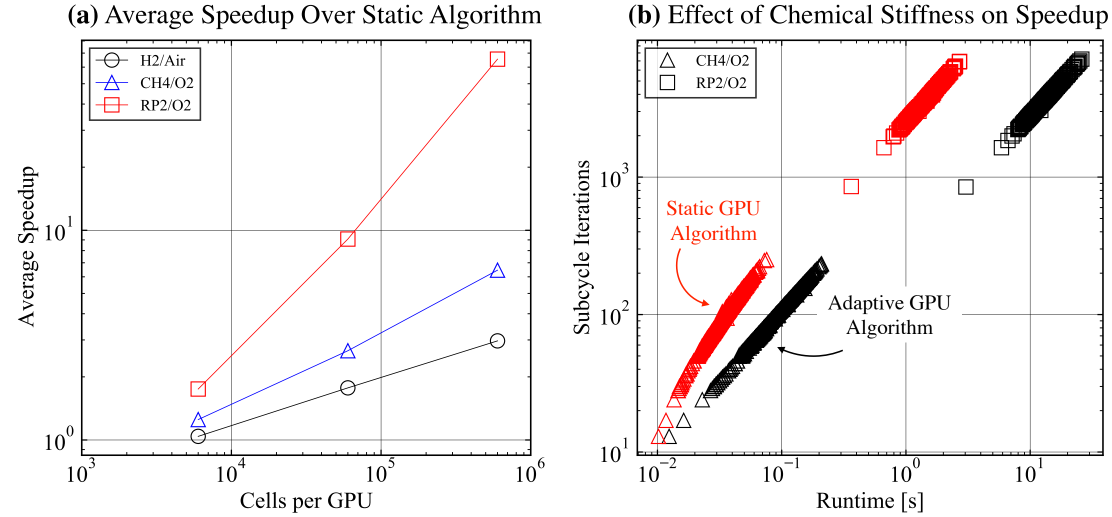


Figure C.6: **(a)** Speedup on average provided by the adaptive time integration (Alg. 5) over the static algorithm (Alg. 4) as a function of N_C. **(b)** Correlation between subcycle (while loop) iterations and runtime (time-to-solution) for the chemical time integration phase.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Abdulle, A., E. Weinan, B. Engquist, and E. Vanden-Eijnden (2012), The heterogeneous multiscale method, *Acta Numerica*, *21*, 1–87.

[2] Adrian, R. (1990), Stochastic estimation of sub-grid scale mations, *Applied Mechanics Reviews*, *43*(5), S214–S218.

[3] Akbari, G., and N. Montazerin (2021), Reconstruction of particle image velocimetry data using flow-based features and validation index: a machine learning approach, *Measurement Science and Technology*, *33*(1), 015,203.

[4] Akram, M. (2021), Reduced-order modeling of turbulent reacting flows using inertial manifold theory, Ph.D. thesis, University of Michigan, Ann Arbor.

[5] Akram, M., and V. Raman (2021), Using approximate inertial manifold approach to model turbulent non-premixed combustion, *Physics of Fluids*, *33*(3), 035,125.

[6] Akram, M., M. Hassanaly, and V. Raman (2020), A priori analysis of reduced description of dynamical systems using approximate inertial manifolds, *Journal of Computational Physics*, *409*, 109,344.

[7] Alexander, F., et al. (2020), Exascale applications: skin in the game, *Philosophical Transactions of the Royal Society A*, *378*(2166), 20190,056.

[8] An, Q., and A. M. Steinberg (2019), The role of strain rate, local extinction, and hydrodynamic instability on transition between attached and lifted swirl flames, *Combustion and Flame*, *199*, 267–278.

[9] An, Q., B. D. Geraedts, and A. M. Steinberg (2015), Dynamics of flame lift-off in biogas swirl flames, in *51st AIAA/SAE/ASEE Joint Propulsion Conference*, p. 4084.

[10] An, Q., W. Y. Kwong, B. D. Geraedts, and A. M. Steinberg (2016), Coupled dynamics of lift-off and precessing vortex core formation in swirl flames, *Combustion and Flame*, *168*, 228–239.

[11] Ananthkrishnan, N., S. Deo, and F. E. Culick (2005), Reduced-order modeling and dynamics of nonlinear acoustic waves in a combustion chamber, *Combustion Science and Technology*, *177*(2), 221–248.

[12] Argonne Leadership Computing Facility (2022), `https://www.alcf.anl.gov/aurora/`.

[13] Arthur, D., and S. Vassilvitskii (2006), k-means++: The advantages of careful seeding, *Tech. rep.*, Stanford.

[14] Athmanathan, V., J. Braun, Z. M. Ayers, C. A. Fugger, A. M. Webb, M. N. Slipchenko, G. Paniagua, S. Roy, and T. R. Meyer (2022), On the effects of reactant stratification and wall curvature in non-premixed rotating detonation combustors, *Combustion and Flame*, *240*, 112,013.

[15] Balci, O. (1995), Principles and techniques of simulation validation, verification, and testing, in *Proceedings of the 27th conference on Winter simulation*, pp. 147–154.

[16] Barwey, S., and V. Raman (2021), A neural network-inspired matrix formulation of chemical kinetics for acceleration on gpus, *Energies*, *14*(9), 2710.

[17] Barwey, S., and V. Raman (2021), Impact of operator splitting schemes on detonation convergence, in *APS Division of Fluid Dynamics Meeting Abstracts*, pp. Q03–001.

[18] Barwey, S., M. Hassanaly, Q. An, V. Raman, and A. Steinberg (2019), Experimental data-based reduced-order model for analysis and prediction of flame transition in gas turbine combustors, *Combustion Theory and Modelling*, *23*(6), 994–1020.

[19] Barwey, S., H. Ganesh, M. Hassanaly, V. Raman, and S. Ceccio (2020), Data-based analysis of multimodal partial cavity shedding dynamics, *Experiments in Fluids*, *61*(4), 1–21.

[20] Barwey, S., S. Prakash, M. Hassanaly, and V. Raman (2021), Data-driven classification and modeling of combustion regimes in detonation waves, *Flow, Turbulence and Combustion*, *106*(4), 1065–1089.

[21] Barwey, S., V. Raman, and A. M. Steinberg (2021), Extracting information overlap in simultaneous oh-plif and piv fields with neural networks, *Proceedings of the Combustion Institute*, *38*(4), 6241–6249.

[22] Barwey, S., M. Hassanaly, V. Raman, and A. Steinberg (2022), Using machine learning to construct velocity fields from oh-plif images, *Combustion Science and Technology*, *194*(1), 93–116.

[23] Batten, P., N. Clarke, C. Lambert, and D. M. Causon (1997), On the choice of wavespeeds for the hllc riemann solver, *SIAM Journal on Scientific Computing*, *18*(6), 1553–1570.

[24] Bauer, M., S. Treichler, E. Slaughter, and A. Aiken (2012), Legion: Expressing locality and independence with logical regions, in *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–11, IEEE.

[25] Beckingsale, D. A., et al. (2019), RAJA: Portable performance for large-scale scientific applications, in *2019 ieee/acm international workshop on performance, portability and productivity in hpc (p3hpc)*, pp. 71–81, IEEE.

[26] Bell, J. B., N. J. Brown, M. S. Day, M. Frenklach, J. F. Grcar, R. M. Propp, and S. R. Tonse (2000), Scaling and efficiency of PRISM in adaptive simulations of turbulent premixed flames, in *Proceedings of the Combustion Institute*, vol. 28, pp. 107–113.

[27] Berger, M. J., and P. Colella (1989), Local adaptive mesh refinement for shock hydrodynamics, *Journal of computational Physics*, *82*(1), 64–84.

[28] Berger, M. J., and J. Oliger (1984), Adaptive mesh refinement for hyperbolic partial differential equations, *Journal of computational Physics*, *53*(3), 484–512.

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

[30] Bielawski, R., S. Barwey, S. Prakash, and V. Raman (In Preparation), Highly-scalable gpu-accelerated compressible reacting solver for shock-containing flows, *Computers & Fluids*.

[31] Blasco, J., N. Fueyo, C. Dopazo, and J. Ballester (1998), Modelling the temporal evolution of a reduced combustion chemical system with an artificial neural network, *Combustion and Flame*, *113*(1-2), 38–52.

[32] Blasco, J. A., N. Fueyo, J. Larroya, C. Dopazo, and Y.-J. Chen (1999), A single-step time-integrator of a methane–air chemical system using artificial neural networks, *Computers & Chemical Engineering*, *23*(9), 1127–1133.

[33] Blasco, J. A., N. Fueyo, C. Dopazo, and J. Chen (2000), A self-organizing-map approach to chemistry representation in combustion applications, *Combustion Theory and Modelling*, *4*(1), 61.

[34] Bode, M., M. Gauding, Z. Lian, D. Denker, M. Davidovic, K. Kleinheinz, J. Jitsev, and H. Pitsch (2021), Using physics-informed enhanced super-resolution generative adversarial networks for subfilter modeling in turbulent reactive flows, *Proceedings of the Combustion Institute*, *38*(2), 2617–2625.

[35] Bogdanoff, D. W. (1994), Advanced injection and mixing techniques for scramjet combustors, *Journal of Propulsion and Power*, *10*(2), 183–190.

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

[37] Bottou, L., and Y. Bengio (1994), Convergence properties of the k-means algorithms, *Advances in neural information processing systems*, *7*.

[38] Bouchacourt, D., R. Tomioka, and S. Nowozin (2018), Multi-level variational autoencoder: Learning disentangled representations from grouped observations, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32.

[39] Brandt, A., and N. Dinar (1979), Multigrid solutions to elliptic flow problems, in *Numerical methods for partial differential equations*, pp. 53–147, Elsevier.

[40] Breuer, K. S., and L. Sirovich (1991), The use of the karhunen-loeve procedure for the calculation of linear eigenfunctions, *Journal of Computational Physics*, *96*(2), 277–296.

[41] Brunton, S. L., B. R. Noack, and P. Koumoutsakos (2020), Machine learning for fluid mechanics, *Annual Review of Fluid Mechanics*, *52*(1), 477–508, doi: 10.1146/annurev-fluid-010719-060214.

[42] Budanur, N. B., K. Y. Short, M. Farazmand, A. P. Willis, and P. Cvitanović (2017), Relative periodic orbits form the backbone of turbulent pipe flow, *Journal of Fluid Mechanics*, *833*, 274–301.

[43] Bulat, G., W. Jones, and A. Marquis (2013), Large eddy simulation of an industrial gas-turbine combustion chamber using the sub-grid pdf method, *Proceedings of the combustion institute*, *34*(2), 3155–3164.

[44] Burkardt, J., M. Gunzburger, and H.-C. Lee (2006), Centroidal voronoi tessellation-based reduced-order modeling of complex systems, *SIAM Journal on Scientific Computing*, *28*(2), 459–484.

[45] Burke, M. P., M. Chaos, Y. Ju, F. L. Dryer, and S. J. Klippenstein (2012), Comprehensive H2/O2 kinetic model for high-pressure combustion, *International Journal of Chemical Kinetics*, *44*(7), 444–474.

[46] Cailler, M., N. Darabiha, D. Veynante, and B. Fiorina (2017), Building-up virtual optimized mechanism for flame modeling, *Proceedings of the Combustion Institute*, *36*(1), 1251 – 1258, doi:10.1016/j.proci.2016.05.028.

[47] Cao, Y., E. Kaiser, J. Borée, B. R. Noack, L. Thomas, and S. Guilain (2014), Cluster-based analysis of cycle-to-cycle variations: application to internal combustion engines, *Experiments in fluids*, *55*(11), 1–8.

[48] Capecelatro, J., D. J. Bodony, and J. Freund (2017), Adjoint-based sensitivity analysis of ignition in a turbulent reactive shear layer, in *55th AIAA Aerospace Sciences Meeting*, p. 0846.

[49] Caron, M., P. Bojanowski, A. Joulin, and M. Douze (2018), Deep clustering for unsupervised learning of visual features, in *Proceedings of the European conference on computer vision (ECCV)*, pp. 132–149.

[50] Caux-Brisebois, V., A. M. Steinberg, C. M. Arndt, and W. Meier (2014), Thermo-acoustic velocity coupling in a swirl stabilized gas turbine model combustor, *Combustion and Flame*, *161*(12), 3166–3180.

[51] Celebi, M. E. (2014), *Partitional clustering algorithms*, Springer.

[52] Chacon, F., and M. Gamba (2019), Study of parasitic combustion in an optically accessible continuous wave rotating detonation engine, in *AIAA Scitech 2019 Forum*, p. 0473.

[53] Chacon, F., A. Feleo, and M. Gamba (2019), Effect of parasitic and commensal combustion on rotating detonation combustor properties, in *Proceedings of the 27th International Colloquium on the Dynamics of Explosions and Reactive Systems*.

[54] Chandrashekar, P. (2013), Kinetic energy preserving and entropy stable finite volume schemes for compressible euler and navier-stokes equations, *Communications in Computational Physics*, *14*(5), 1252–1286.

[55] Chen, J. H., et al. (2009), Terascale direct numerical simulations of turbulent combustion using s3d, *Computational Science & Discovery*, *2*(1), 015,001.

[56] Chen, K. K., J. H. Tu, and C. W. Rowley (2012), Variants of dynamic mode decomposition: boundary condition, koopman, and fourier analyses, *Journal of nonlinear science*, *22*(6), 887–915.

[57] Chiavazzo, E., C. W. Gear, C. J. Dsilva, N. Rabin, and I. G. Kevrekidis (2014), Reduced models in chemical kinetics via nonlinear data-mining, *Processes*, *2*(1), 112–140.

[58] Chong, S. T., M. Hassanaly, H. Koo, M. E. Mueller, V. Raman, and K.-P. Geigle (2018), Large eddy simulation of pressure and dilution-jet effects on soot formation in a model aircraft swirl combustor, *Combustion and Flame*, *192*, 452–472.

[59] Choubey, G., Y. Devarajan, W. Huang, K. Mehar, M. Tiwari, and K. Pandey (2019), Recent advances in cavity-based scramjet engine-a brief review, *International Journal of Hydrogen Energy*, *44*(26), 13,895–13,909.

[60] Christo, F., A. Masri, and E. Nebot (1996), Artificial neural network implementation of chemistry with pdf simulation of h2/co2 flames, *Combustion and Flame*, *106*(4), 406–427.

[61] Chterev, I., et al. (2014), Flame and flow topologies in an annular swirling flow, *Combustion Science and Technology*, *186*(8), 1041–1074.

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

[63] Cohen, S. D., A. C. Hindmarsh, and P. F. Dubois (1996), Cvode, a stiff/nonstiff ode solver in c, *Computers in physics*, *10*(2), 138–143.

[64] Colella, P., A. Majda, and V. Roytburd (1986), Theoretical and numerical structure for reacting shock waves, *SIAM Journal on Scientific and Statistical Computing*, *7*(4), 1059–1080.

[65] Courant, R., K. Friedrichs, and H. Lewy (1928), Über die partiellen differenzengleichungen der mathematischen physik, *Mathematische annalen*, *100*(1), 32–74.

[66] Coussement, A., O. Gicquel, and A. Parente (2013), Mg-local-pca method for reduced order combustion modeling, *Proceedings of the Combustion Institute*, *34*(1), 1117–1123.

[67] Curran, E. T. (2001), Scramjet engines: the first forty years, *Journal of Propulsion and Power*, *17*(6), 1138–1148.

[68] Curtis, N. J., K. E. Niemeyer, and C.-J. Sung (2018), Using simd and simt vectorization to evaluate sparse chemical kinetic jacobian matrices and thermochemical source terms, *Combustion and Flame*, *198*, 186–204.

[69] Cvitanović, P., and B. Eckhardt (1993), Symmetry decomposition of chaotic dynamics, *Nonlinearity*, *6*(2), 277.

[70] Deiterding, R. (2003), *Parallel adaptive simulation of multi-dimensional detonation structures*, Dissertation. de.

[71] Deng, Z., C. He, Y. Liu, and K. C. Kim (2019), Super-resolution reconstruction of turbulent velocity fields using a generative adversarial network-based artificial intelligence framework, *Physics of Fluids*, *31*(12), 125,111.

[72] Dhillon, I. S., and D. S. Modha (2002), A data-clustering algorithm on distributed memory multiprocessors, in *Large-scale parallel data mining*, pp. 245–260, Springer.

[73] Dhillon, I. S., Y. Guan, and B. Kulis (2004), Kernel k-means: spectral clustering and normalized cuts, in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 551–556.

[74] Ding, C., and X. He (2004), K-means clustering via principal component analysis, in *Proceedings of the twenty-first international conference on Machine learning*, p. 29.

[75] Dosovitskiy, A., P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox (2015), Flownet: Learning optical flow with convolutional networks, in *Proceedings of the IEEE international conference on computer vision*, pp. 2758–2766.

[76] Driver, H. E., and A. L. Kroeber (1932), *Quantitative expression of cultural relationships*, vol. 31, Berkeley: University of California Press.

[77] Drmac, Z., I. Mezic, and R. Mohr (2018), Data driven modal decompositions: analysis and enhancements, *SIAM Journal on Scientific Computing*, *40*(4), A2253–A2285.

[78] Duraisamy, K. (2021), Perspectives on machine learning-augmented reynolds-averaged and large eddy simulation models of turbulence, *Physical Review Fluids*, *6*(5), 050,504.

[79] Duraisamy, K., G. Iaccarino, and H. Xiao (2019), Turbulence modeling in the age of data, *Annual Review of Fluid Mechanics*, *51*, 357–377.

[80] D'Alessio, G., A. Parente, A. Stagni, and A. Cuoci (2020), Adaptive chemistry via pre-partitioning of composition space and mechanism reduction, *Combustion and Flame*, *211*, 68–82.

[81] Edwards, H. C., C. R. Trott, and D. Sunderland (2014), Kokkos: Enabling manycore performance portability through polymorphic memory access patterns, *Journal of parallel and distributed computing*, *74*(12), 3202–3216.

[82] Emmett, M., E. Motheau, W. Zhang, M. Minion, and J. B. Bell (2019), A fourth-order adaptive mesh refinement algorithm for the multicomponent, reacting compressible navier–stokes equations, *Combustion Theory and Modelling*, *23*(4), 592–625.

[83] Farazmand, M., and T. P. Sapsis (2016), Dynamical indicators for the prediction of bursting phenomena in high-dimensional systems, *Physical Review E*, *94*(3), 032,212.

[84] Fard, M. M., T. Thonet, and E. Gaussier (2020), Deep k-means: Jointly clustering with k-means and learning representations, *Pattern Recognition Letters*, *138*, 185–192.

[85] Farivar, R., D. Rebolledo, E. Chan, and R. H. Campbell (2008), A parallel implementation of k-means clustering on gpus., in *Pdpta*, vol. 13, pp. 212–312.

[86] Fickett, W., and W. C. Davis (2000), *Detonation: theory and experiment*, Courier Corporation.

[87] Fiévet, R., H. Koo, and V. Raman (2015), Numerical simulation of a scramjet isolator with thermodynamic nonequilibrium, in *22nd AIAA Computational Fluid Dynamics Conference*, p. 3418, doi:10.2514/6.2015-3418.

[88] Fiévet, R., H. Koo, V. Raman, and A. H. Auslender (2017), Numerical investigation of shock-train response to inflow boundary-layer variations, *AIAA Journal*, *55*(9), 2888–2901.

[89] Fiévet, R., S. Voelkel, H. Koo, V. Raman, and P. L. Varghese (2017), Effect of thermal nonequilibrium on ignition in scramjet combustors, *Proceedings of the Combustion Institute*, *36*(2), 2901–2910.

[90] Fiorina, B., and M. Caillier (2019), Accounting for complex chemistry in the simulations of future turbulent combustion systems, in *AIAA SciTech 2019 Forum*, p. 0995.

[91] Fogleman, M., J. Lumley, D. Rempfer, and D. Haworth (2004), Application of the proper orthogonal decomposition to datasets of internal combustion engine flows, *Journal of Turbulence*, *5*(1), 023.

[92] Foias, C., G. R. Sell, and R. Temam (1988), Inertial manifolds for nonlinear evolutionary equations, *Journal of differential equations*, *73*(2), 309–353.

[93] Fooladgar, E., and C. Duwig (2018), A new post-processing technique for analyzing high-dimensional combustion data, *Combustion and Flame*, *191*, 226–238.

[94] Fornberg, B., and D. M. Sloan (1994), A review of pseudospectral methods for solving partial differential equations, *Acta numerica*, *3*, 203–267.

[95] Fotia, M. L., and J. F. Driscoll (2013), Ram-scram transition and flame/shock-train interactions in a model scramjet experiment, *Journal of Propulsion and Power*, *29*(1), 261–273.

[96] Franke, L. L., A. K. Chatzopoulos, and S. Rigopoulos (2017), Tabulation of combustion chemistry via artificial neural networks (ANNs): Methodology and application to LES-PDF simulation of Sydney flame L, *Combustion and Flame*, *185*, 245–260.

[97] Frolov, S., A. Dubrovskii, and V. Ivanov (2013), Three-dimensional numerical simulation of operation process in rotating detonation engine, *Progress in Propulsion Physics*, *4*, 467–488.

[98] Fry, R. S. (2004), A century of ramjet propulsion technology evolution, *Journal of propulsion and power*, *20*(1), 27–58.

[99] Fukami, K., K. Fukagata, and K. Taira (2019), Super-resolution analysis with machine learning for low-resolution flow data, in *11th International Symposium on Turbulence and Shear Flow Phenomena, TSFP 2019.*

[100] Fukami, K., T. Nakamura, and K. Fukagata (2020), Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data, *Physics of Fluids*, *32*(9), 095,110.

[101] Galassi, R. M., P. P. Ciottoli, M. Valorani, and H. G. Im (2022), An adaptive time-integration scheme for stiff chemistry based on computational singular perturbation and artificial neural networks, *Journal of Computational Physics*, *451*, 110,875.

[102] Garicano-Mena, J., B. Li, E. Ferrer, and E. Valero (2019), A composite dynamic mode decomposition analysis of turbulent channel flows, *Physics of Fluids*, *31*(11), 115,102.

[103] Gaspard, P., G. Nicolis, A. Provata, and S. Tasaki (1995), Spectral signature of the pitchfork bifurcation: Liouville equation approach, *Physical Review E*, *51*(1), 74.

[104] Gibney, E., et al. (2021), Fuel for world's largest fusion reactor iter is set for test run, *Nature*, *591*(7848), 15–16.

[105] Gicquel, L. Y., G. Staffelbach, and T. Poinsot (2012), Large eddy simulations of gaseous flames in gas turbine combustion chambers, *Progress in energy and combustion science*, *38*(6), 782–817.

[106] Glassman, I., R. A. Yetter, and N. G. Glumac (2014), *Combustion*, Academic press.

[107] Glaws, A., R. King, and M. Sprague (2020), Deep learning for in situ data compression of large turbulent flow simulations, *Physical Review Fluids*, *5*(11), 114,602.

[108] Goodfellow, I., Y. Bengio, and A. Courville (2016), *Deep learning*, MIT press.

[109] Goodwin, D. G., H. K. Moffat, I. Schoegl, R. L. Speth, and B. W. Weber (2022), Cantera: An object-oriented software toolkit for chemical kinetics, thermodynamics, and transport processes.

[110] Gottlieb, S., D. I. Ketcheson, and C.-W. Shu (2009), High order strong stability preserving time discretizations, *Journal of Scientific Computing*, *38*(3), 251–289.

[111] Goussis, D. A., and M. Valorani (2006), An efficient iterative algorithm for the approximation of the fast and slow dynamics of stiff systems, *Journal of Computational Physics*, *214*(1), 316–346.

[112] Graftieaux, L., M. Michard, and N. Grosjean (2001), Combining PIV, POD and vortex identification algorithms for the study of unsteady turbulent swirling flows, *Measurement Science and Technology*, *12*(9), 1422.

[113] Gruber, A., M. Gunzburger, L. Ju, and Z. Wang (2022), A comparison of neural network architectures for data-driven reduced-order modeling, *Computer Methods in Applied Mechanics and Engineering*, *393*, 114,764.

[114] Gui, M.-Y., B.-C. Fan, and G. Dong (2011), Periodic oscillation and fine structure of wedge-induced oblique detonation waves, *Acta Mechanica Sinica*, *27*(6), 922–928.

[115] Hadjighasem, A., D. Karrasch, H. Teramoto, and G. Haller (2016), Spectral-clustering approach to lagrangian vortex detection, *Physical Review E*, *93*(6), 063,107.

[116] Han, W., V. Raman, M. E. Mueller, and Z. Chen (2019), Effects of combustion models on soot formation and evolution in turbulent nonpremixed flames, *Proceedings of the Combustion Institute*, *37*(1), 985–992.

[117] Hansinger, M., Y. Ge, and M. Pfitzner (2022), Deep residual networks for flamelet/progress variable tabulation with application to a piloted flame with inhomogeneous inlet, *Combustion Science and Technology*, *194*(8), 1587–1613.

[118] Hardoon, D. R., S. Szedmak, and J. Shawe-Taylor (2004), Canonical correlation analysis: An overview with application to learning methods, *Neural computation*, *16*(12), 2639–2664.

[119] Hassanaly, M. (2020), Extreme events in turbulent combustion, Ph.D. thesis, University of Michigan, Ann Arbor.

[120] Hassanaly, M., and V. Raman (2019), Ensemble-les analysis of perturbation response of turbulent partially-premixed flames, *Proceedings of the Combustion Institute*, *37*(2), 2249–2257.

[121] Hassanaly, M., and V. Raman (2019), Lyapunov spectrum of forced homogeneous isotropic turbulent flows, *Physical Review Fluids*, *4*(11), 114,608.

[122] Hassanaly, M., and V. Raman (2021), Classification and computation of extreme events in turbulent combustion, *Progress in Energy and Combustion Science*, *87*, 100,955.

[123] Hassanaly, M., Y. Tang, S. Barwey, and V. Raman (2021), Data-driven analysis of relight variability of jet fuels induced by turbulence, *Combustion and Flame*, *225*, 453–467.

[124] Hassanaly, M., A. Glaws, K. Stengel, and R. N. King (2022), Adversarial sampling of unknown and high-dimensional conditional distributions, *Journal of Computational Physics*, *450*, 110,853.

[125] Hecht-Nielsen, R. (1992), Theory of the backpropagation neural network, in *Neural Networks for Perception*, pp. 65–93, Elsevier.

[126] Herrmann, M., G. Blanquart, and V. Raman (2006), Flux corrected finite volume scheme for preserving scalar boundedness in reacting large-eddy simulations, *AIAA journal*, *44*(12), 2879–2886.

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

[128] Holmes, P., J. L. Lumley, G. Berkooz, and C. W. Rowley (2012), *Turbulence, coherent structures, dynamical systems and symmetry*, Cambridge university press.

[129] Hooker, S. (2021), The hardware lottery, *Communications of the ACM*, *64*(12), 58–65.

[130] Hooker, S. (2021), The hardware lottery, *Communications of the ACM*, *64*(12), 58–65.

[131] Huang, J. Z., M. K. Ng, H. Rong, and Z. Li (2005), Automated variable weighting in k-means type clustering, *IEEE transactions on pattern analysis and machine intelligence*, *27*(5), 657–668.

[132] Huang, Y., and V. Yang (2009), Dynamics and stability of lean-premixed swirl-stabilized combustion, *Progress in energy and combustion science*, *35*(4), 293–364.

[133] Ihme, M., W. T. Chung, and A. A. Mishra (2022), Combustion machine learning: Principles, progress and prospects, *Progress in Energy and Combustion Science*, *91*, 101,010.

[134] Iudiciani, P., C. Duwig, S. Husseini, R.-Z. Szasz, L. Fuchs, and E. Gutmark (2012), Proper orthogonal decomposition for experimental investigation of flame instabilities, *AIAA journal*, *50*(9), 1843–1854.

[135] Jacobsen, C., and K. Duraisamy (2022), Disentangling generative factors of physical fields using variational autoencoders, *Frontiers in Physics*, p. 536.

[136] Jacobsen, L. S., C. D. Carter, T. A. Jackson, S. Williams, J. Barnett, D. Bivolaru, S. Kuo, C.-J. Tam, and R. A. Baurle (2008), Plasma-assisted ignition in scramjets, *Journal of Propulsion and Power*, *24*(4), 641–654.

[137] Jain, A. K., M. N. Murty, and P. J. Flynn (1999), Data clustering: a review, *ACM computing surveys (CSUR)*, *31*(3), 264–323.

[138] Jameson, A. (1983), Solution of the euler equations for two dimensional transonic flow by a multigrid method, *Applied mathematics and computation*, *13*(3-4), 327–355.

[139] Jiang, G., and D. Peng (2000), Weighted eno schemes for hamilton–jacobi equations, *SIAM Journal on Scientific Computing*, *21*(6), 2126–2143, doi: 10.1137/S106482759732455X.

[140] Kaiser, E., et al. (2014), Cluster-based reduced-order modelling of a mixing layer, *Journal of Fluid Mechanics*, *754*, 365–414.

[141] Karnakov, P., S. Litvinov, and P. Koumoutsakos (2022), Optimizing a discrete loss (odil) to solve forward and inverse problems for partial differential equations using machine learning tools, *arXiv preprint arXiv:2205.04611*.

[142] Kaul, C. M., V. Raman, E. Knudsen, E. S. Richardson, and J. H. Chen (2013), Large eddy simulation of a lifted ethylene flame using a dynamic nonequilibrium model for subfilter scalar variance and dissipation rate, *Proceedings of the Combustion Institute*, *34*(1), 1289–1297.

[143] Kawai, S., and S. K. Lele (2010), Large-eddy simulation of jet mixing in supersonic crossflows, *AIAA journal*, *48*(9), 2063–2083.

[144] Kee, R. J., F. M. Rupley, and J. A. Miller (1989), Chemkin-ii: A fortran chemical kinetics package for the analysis of gas-phase chemical kinetics, *Tech. rep.*, Sandia National Lab.(SNL-CA), Livermore, CA (United States).

[145] Kempf, A., F. Flemming, and J. Janicka (2005), Investigation of lengthscales, scalar dissipation, and flame orientation in a piloted diffusion flame by LES, *Proceedings of the Combustion Institute*, *30*(1), 557–565.

[146] Kerdprasop, K., N. Kerdprasop, and P. Sattayatham (2005), Weighted k-means for density-biased clustering, in *International conference on data warehousing and knowledge discovery*, pp. 488–497, Springer.

[147] Kevrekidis, I. G., C. W. Gear, and G. Hummer (2004), Equation-free: The computer-aided analysis of complex multiscale systems, *AIChE Journal*, *50*(7), 1346–1355.

[148] Kim, H., J. Kim, S. Won, and C. Lee (2021), Unsupervised deep learning for super-resolution reconstruction of turbulence, *Journal of Fluid Mechanics*, *910*.

[149] Kim, K., O. Diaz-Ibarra, C. Safta, and H. Najm (2021), TINES - Time Integration, Newton and Eigen Solver, *Sandia Report*, *0*, 0.

[150] Kim, S., W. Ji, S. Deng, Y. Ma, and C. Rackauckas (2021), Stiff neural ordinary differential equations, *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *31*(9), 093,122.

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

[152] Kingma, D. P., M. Welling, et al. (2019), An introduction to variational autoencoders, *Foundations and Trends® in Machine Learning*, *12*(4), 307–392.

[153] Klein, M., A. Sadiki, and J. Janicka (2003), A digital filter based generation of inflow data for spatially developing direct numerical or large eddy simulations, *Journal of computational Physics*, *186*(2), 652–665.

[154] Klus, S., P. Koltai, and C. Schütte (2015), On the numerical approximation of the perron-frobenius and koopman operator, *arXiv preprint arXiv:1512.05997*.

[155] Knudsen, E., H. Pitsch, et al. (2015), Modeling partially premixed combustion behavior in multiphase les, *Combustion and Flame*, *162*(1), 159–180.

[156] Kochkov, D., J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer (2021), Machine learning–accelerated computational fluid dynamics, *Proceedings of the National Academy of Sciences*, *118*(21), e2101784,118.

[157] Kodinariya, T. M., and P. R. Makwana (2013), Review on determining number of cluster in k-means clustering, *International Journal*, *1*(6), 90–95.

[158] Kong, C., J.-T. Chang, Y.-F. Li, and R.-Y. Chen (2020), Deep learning methods for super-resolution reconstruction of temperature fields in a supersonic combustor, *AIP Advances*, *10*(11), 115,021.

[159] Koo, H., V. Raman, and P. L. Varghese (2015), Direct numerical simulation of supersonic combustion with thermal nonequilibrium, *Proceedings of the Combustion Institute*, *35*(2), 2145–2153.

[160] Kramer, M. A. (1991), Nonlinear principal component analysis using autoassociative neural networks, *AIChE journal*, *37*(2), 233–243.

[161] Kunisch, K., and S. Volkwein (2001), Galerkin proper orthogonal decomposition methods for parabolic problems, *Numerische mathematik*, *90*(1), 117–148.

[162] Kutz, J. N. (2013), *Data-driven modeling & scientific computation: methods for complex systems & big data*, Oxford University Press.

[163] Ladeinde, F. (2010), Advanced computational-fluid-dynamics techniques for scramjet combustion simulation, *Aiaa Journal*, *48*(3), 513–514.

[164] Lam, S.-H., and D. A. Goussis (1989), Understanding complex chemical kinetics with computational singular perturbation, in *Symposium (International) on Combustion*, vol. 22, pp. 931–941, Elsevier.

[165] Langford, J. A., and R. D. Moser (1999), Optimal les formulations for isotropic turbulence, *Journal of fluid mechanics*, *398*, 321–346.

[166] Lanser, D., and J. G. Verwer (1999), Analysis of operator splitting for advection–diffusion–reaction problems from air pollution modelling, *Journal of computational and applied mathematics*, *111*(1-2), 201–216.

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

[168] Larsson, J., R. Vicquelin, and I. Bermejo-Moreno (2011), Large eddy simulations of the hyshot ii scramjet, *Center for Turbulence Research Annual Briefs*, pp. 63–74.

[169] Lee, K., and K. T. Carlberg (2020), Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders, *Journal of Computational Physics*, *404*, 108,973.

[170] Lee, K., and E. J. Parish (2021), Parameterized neural ordinary differential equations: Applications to computational physics problems, *Proceedings of the Royal Society A*, *477*(2253), 20210,162.

[171] Lee, S., D.-R. Cho, and J.-Y. Choi (2008), Effect of curvature on the detonation wave propagation characteristics in annular channels, in *46th AIAA Aerospace Sciences Meeting and Exhibit*, p. 988.

[172] Lee, S., S.-J. Min, and R. Eigenmann (2009), OpenMP to GPGPU: a compiler framework for automatic translation and optimization, *ACM Sigplan Notices*, *44*(4), 101–110.

[173] Lefebvre, M., and E. Oran (1995), Analysis of the shock structures in a regular detonation, *Shock Waves*, *4*(5), 277–283.

[174] Lei, C., J. Deng, K. Cao, L. Ma, Y. Xiao, and L. Ren (2018), A random forest approach for predicting coal spontaneous combustion, *Fuel*, *223*, 63–73.

[175] LeVeque, R. J., et al. (2002), *Finite volume methods for hyperbolic problems*, vol. 31, Cambridge university press.

[176] Li, H., D. Fernex, R. Semaan, J. Tan, M. Morzyński, and B. R. Noack (2021), Cluster-based network model, *Journal of Fluid Mechanics*, *906*.

[177] Li, Q., and Z. Wang (2017), Dynamic mode decomposition of turbulent combustion process in dlr scramjet combustor, *Journal of Aerospace Engineering*, *30*(5), 04017,034.

[178] Liao, T. W. (2005), Clustering of time series data—a survey, *Pattern recognition*, *38*(11), 1857–1874.

[179] Liberge, E., and A. Hamdouni (2010), Reduced order modelling method via proper orthogonal decomposition (pod) for flow around an oscillating cylinder, *Journal of fluids and structures*, *26*(2), 292–311.

[180] Lorenz, E. N. (1963), Deterministic nonperiodic flow, *Journal of atmospheric sciences*, *20*(2), 130–141.

[181] Lu, F. K., and E. M. Braun (2014), Rotating detonation wave propulsion: experimental challenges, modeling, and engine concepts, *Journal of Propulsion and Power*, *30*(5), 1125–1142.

[182] Lu, T., and C. K. Law (2005), A directed relation graph method for mechanism reduction, *Proceedings of the Combustion Institute*, *30*(1), 1333–1341.

[183] Lu, T., and C. K. Law (2009), Toward accommodating realistic fuel chemistry in large-scale computations, *Progress in Energy and Combustion Science*, *35*(2), 192–215.

[184] Lu, Z., H. Zhou, S. Li, Z. Ren, T. Lu, and C. K. Law (2017), Analysis of operator splitting errors for near-limit flame simulations, *Journal of Computational Physics*, *335*, 578–591.

[185] Lumley, J. L., and A. Poje (1997), Low-dimensional models for flows with density fluctuations, *Physics of Fluids*, *9*(7), 2023–2031.

[186] Lund, T. (2003), The use of explicit filters in large eddy simulation, *Computers & Mathematics with Applications*, *46*(4), 603–616.

[187] Lusch, B., J. N. Kutz, and S. L. Brunton (2018), Deep learning for universal linear embeddings of nonlinear dynamics, *Nature communications*, *9*(1), 1–10.

[188] Lv, Y., and M. Ihme (2014), Discontinuous galerkin method for multicomponent chemically reacting flows and combustion, *Journal of Computational Physics*, *270*, 105–137.

[189] Ma, F., J. Li, V. Yang, K.-C. Lin, and T. Jackson (2005), Thermoacoustic flow instability in a scramjet combustor, in *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, p. 3824.

[190] Maas, U., and S. B. Pope (1992), Simplifying chemical kinetics: intrinsic low-dimensional manifolds in composition space, *Combustion and flame*, *88*(3-4), 239–264.

[191] Mahmoudi, Y., and K. Mazaheri (2011), High resolution numerical simulation of the structure of 2-d gaseous detonations, *Proceedings of the Combustion Institute*, *33*(2), 2187–2194.

[192] Malik, M. R., B. J. Isaac, A. Coussement, P. J. Smith, and A. Parente (2018), Principal component analysis coupled with nonlinear regression for chemistry reduction, *Combustion and Flame*, *187*, 30–41.

[193] Marion, M., and R. Temam (1989), Nonlinear galerkin methods, *SIAM Journal on numerical analysis*, *26*(5), 1139–1157.

[194] Markovich, D., S. Abdurakipov, L. Chikishev, V. Dulin, and K. Hanjalić (2014), Comparative analysis of low-and high-swirl confined flames and jets by proper orthogonal and dynamic mode decompositions, *Physics of Fluids*, *26*(6), 065,109.

[195] Massa, L., J. Austin, and T. Jackson (2007), Triple-point shear layers in gaseous detonation waves, *Journal of Fluid Mechanics*, *586*, 205–248.

[196] Mathew, J., R. Lechner, H. Foysi, J. Sesterhenn, and R. Friedrich (2003), An explicit filtering method for large eddy simulation of compressible flows, *Physics of fluids*, *15*(8), 2279–2289.

[197] Mathew, J., R. Lechner, H. Foysi, J. Sesterhenn, and R. Friedrich (2003), An explicit filtering method for large eddy simulation of compressible flows, *Physics of fluids*, *15*(8), 2279–2289.

[198] Maulik, R., A. Mohan, B. Lusch, S. Madireddy, P. Balaprakash, and D. Livescu (2020), Time-series learning of latent-space dynamics for reduced-order model closure, *Physica D: Nonlinear Phenomena*, *405*, 132,368.

[199] Maulik, R., B. Lusch, and P. Balaprakash (2021), Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders, *Physics of Fluids*, *33*(3), 037,106.

[200] McCorquodale, P., P. Colella, and H. Johansen (2001), A cartesian grid embedded boundary method for the heat equation on irregular domains, *Journal of Computational Physics*, *173*(2), 620–635.

[201] Mehl, M., W. J. Pitz, C. K. Westbrook, and H. J. Curran (2011), Kinetic modeling of gasoline surrogate components and mixtures under engine conditions, *Proceedings of the Combustion Institute*, *33*(1), 193–200.

[202] Meier, W., P. Weigand, X. Duan, and R. Giezendanner-Thoben (2007), Detailed characterization of the dynamics of thermoacoustic pulsations in a lean premixed swirl flame, *Combustion and Flame*, *150*(1-2), 2–26.

[203] Mescheder, L., S. Nowozin, and A. Geiger (2017), Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks, in *International conference on machine learning*, pp. 2391–2400, PMLR.

[204] Metcalfe, W. K., S. M. Burke, S. S. Ahmed, and H. J. Curran (2013), A hierarchical and comparative kinetic modeling study of c1- c2 hydrocarbon and oxygenated fuels, *International Journal of Chemical Kinetics*, *45*(10), 638–675.

[205] Mezić, I. (2005), Spectral properties of dynamical systems, model reduction and decompositions, *Nonlinear Dynamics*, *41*(1), 309–325.

[206] Mezic, I. (2013), Analysis of fluid flows via spectral properties of the koopman operator, *Annual Review of Fluid Mechanics*, *45*(1), 357–378.

[207] Mittal, S., and J. S. Vetter (2015), A survey of cpu-gpu heterogeneous computing techniques, *ACM Computing Surveys (CSUR)*, *47*(4), 1–35.

[208] Mittal, V., and H. Pitsch (2013), A flamelet model for premixed combustion under variable pressure conditions, *Proceedings of the Combustion Institute*, *34*(2), 2995–3003.

[209] Moin, P., and K. Mahesh (1998), Direct numerical simulation: a tool in turbulence research, *Annual review of fluid mechanics*, *30*(1), 539–578.

[210] Moore, G. E. (1998), Cramming more components onto integrated circuits, *Proceedings of the IEEE*, *86*(1), 82–85.

[211] Motheau, E., F. Nicoud, and T. Poinsot (2014), Mixed acoustic-entropy combustion instabilities in gas turbines, *Journal of Fluid Mechanics*, *749*, 542–576.

[212] Mueller, M., T. Kim, R. Yetter, and F. Dryer (1999), Flow reactor studies and kinetic modeling of the H_2/O_2 reaction, *International journal of chemical kinetics*, *31*(2), 113–125.

[213] Mueller, M. E., and H. Pitsch (2013), Large eddy simulation of soot evolution in an aircraft combustor, *Physics of Fluids*, *25*(11), 110,812.

[214] Murphy, K. P. (2012), *Machine learning: a probabilistic perspective*, MIT press.

[215] Murtagh, F., and P. Contreras (2012), Algorithms for hierarchical clustering: an overview, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(1), 86–97.

[216] Naik, D. S. B., S. D. Kumar, and S. Ramakrishna (2013), Parallel processing of enhanced k-means using openmp, in *2013 IEEE International Conference on Computational Intelligence and Computing Research*, pp. 1–4, IEEE.

[217] Najafi-Yazdi, A., B. Cuenot, and L. Mongeau (2012), Systematic definition of progress variables and intrinsically low-dimensional, flamelet generated manifolds for chemistry tabulation, *Combustion and Flame*, *159*(3), 1197–1204.

[218] Najm, H. N., P. S. Wyckoff, and O. M. Knio (1998), A semi-implicit numerical scheme for reacting flow: I. stiff chemistry, *Journal of Computational Physics*, *143*(2), 381–402.

[219] Nakahashi, K., Y. Ito, and F. Togashi (2003), Some challenges of realistic flow simulations by unstructured grid cfd, *International Journal for Numerical Methods in Fluids*, *43*(6-7), 769–783.

[220] Nguyen, P.-D., L. Vervisch, V. Subramanian, and P. Domingo (2010), Multidimensional flamelet-generated manifolds for partially premixed combustion, *Combustion and Flame*, *157*(1), 43–61.

[221] Niemeyer, K. E., and C.-J. Sung (2014), Accelerating moderately stiff chemical kinetics in reactive-flow simulations using gpus, *Journal of Computational Physics*, *256*, 854–871.

[222] Niemeyer, K. E., and C.-J. Sung (2014), Recent progress and challenges in exploiting graphics processors in computational fluid dynamics, *The Journal of Supercomputing*, *67*(2), 528–564.

[223] Noiray, N., and B. Schuermans (2013), On the dynamic nature of azimuthal thermoacoustic modes in annular gas turbine combustion chambers, *Proc. R. Soc. A*, *469*(2151), 20120,535.

[224] Noiray, N., M. Bothien, and B. Schuermans (2011), Investigation of azimuthal staging concepts in annular gas turbines, *Combustion Theory and Modelling*, *15*(5), 585–606.

[225] Nordeen, C. A., D. Schwer, F. Schauer, J. Hoke, T. Barber, and B. Cetegen (2014), Thermodynamic model of a rotating detonation engine, *Combustion, Explosion, and Shock Waves*, *50*(5), 568–577.

[226] Nordström, J. (2006), Conservative finite difference formulations, variable coefficients, energy estimates and artificial dissipation, *Journal of Scientific Computing*, *29*(3), 375–404.

[227] Oak Ridge Leadership Computing Facility (2018), `https://www.olcf.ornl.gov/summit/`.

[228] Oak Ridge Leadership Computing Facility (2022), `https://www.olcf.ornl.gov/frontier/`.

[229] Oberleithner, K., M. Sieber, C. N. Nayeri, C. O. Paschereit, C. Petz, H.-C. Hege, B. R. Noack, and I. Wygnanski (2011), Three-dimensional coherent structures in a swirling jet undergoing vortex breakdown: stability analysis and empirical mode construction, *Journal of Fluid Mechanics*, *679*, 383–414.

[230] Oberleithner, K., M. Stöhr, S. H. Im, C. M. Arndt, and A. M. Steinberg (2015), Formation and flame-induced suppression of the precessing vortex core in a swirl combustor: experiments and linear stability analysis, *Combustion and Flame*, *162*(8), 3100–3114.

[231] O'Connor, J., and T. Lieuwen (2011), Disturbance field characteristics of a transversely excited burner, *Combustion Science and Technology*, *183*(5), 427–443.

[232] of California at San Diego, U. (2016), Chemical-kinetic mechanisms for combustion applications, `http://web.eng.ucsd.edu/mae/groups/combustion/mechanism.html`, San Diego Mechanism web page, Mechanical and Aerospace Engineering (Combustion Research), University of California at San Diego.

[233] Oke, T. R., G. Mills, A. Christen, and J. A. Voogt (2017), *Urban climates*, Cambridge University Press.

[234] Osher, S. (1985), Convergence of generalized muscl schemes, *SIAM Journal on Numerical Analysis*, *22*(5), 947–961.

[235] Owens, J. D., M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips (2008), Gpu computing, *Proceedings of the IEEE*, *96*(5), 879–899.

[236] Owoyele, O., and P. Pal (2020), Chemnode: A neural ordinary differential equations approach for chemical kinetics solvers, *arXiv preprint arXiv:2101.04749*.

[237] Owoyele, O., P. Kundu, and P. Pal (2021), Efficient bifurcation and tabulation of multi-dimensional combustion manifolds using deep mixture of experts: An a priori study, *Proceedings of the Combustion Institute*, *38*(4), 5889–5896.

[238] Pal, P., C. Xu, G. Kumar, S. A. Drennan, B. A. Rankin, and S. Som (2020), Large-eddy simulations and mode analysis of ethylene/air combustion in a non-premixed rotating detonation engine, in *AIAA Propulsion and Energy 2020 Forum*, p. 3876.

[239] Pan, S., and K. Duraisamy (2020), Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability, *SIAM Journal on Applied Dynamical Systems*, *19*(1), 480–509.

[240] Pan, S., N. Arnold-Medabalimi, and K. Duraisamy (2021), Sparsity-promoting algorithms for the discovery of informative koopman-invariant subspaces, *Journal of Fluid Mechanics*, *917*.

[241] Papatheodore, T. (2018), Summit system overview, `https://www.olcf.ornl.gov/wp-content/uploads/2018/05/Intro_Summit_System_Overview.pdf`.

[242] Paszke, A., et al. (2017), Automatic differentiation in pytorch.

[243] Patera, A. T. (1984), A spectral element method for fluid dynamics: laminar flow in a channel expansion, *Journal of computational Physics*, *54*(3), 468–488.

[244] Patki, T., D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. De Supinski (2013), Exploring hardware overprovisioning in power-constrained, high performance computing, in *Proceedings of the 27th international ACM conference on International conference on supercomputing*, pp. 173–182.

[245] Pecnik, R., V. E. Terrapon, F. Ham, G. Iaccarino, and H. Pitsch (2012), Reynolds-averaged navier-stokes simulations of the hyshot ii scramjet, *AIAA journal*, *50*(8), 1717–1732.

[246] Pelleg, D., A. W. Moore, et al. (2000), X-means: Extending k-means with efficient estimation of the number of clusters., in *Icml*, vol. 1, pp. 727–734.

[247] Pellett, G., C. Bruno, and W. Chinitz (2002), Review of air vitiation effects on scramjet ignition and flameholding combustion processes, in *38th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, p. 3880.

[248] Pember, R. B., L. H. Howell, J. B. Bell, P. Colella, W. Y. Crutchfield, W. Fiveland, and J. Jessee (1998), An adaptive projection method for unsteady, low-mach number combustion, *Combustion Science and Technology*, *140*(1-6), 123–168.

[249] Pérez, F. E. H., N. Mukhadiyev, X. Xu, A. Sow, B. J. Lee, R. Sankaran, and H. G. Im (2018), Direct numerical simulations of reacting flows with detailed chemistry using many-core/gpu acceleration, *Computers & Fluids*, *173*, 73–79.

[250] Peskin, C. S. (2002), The immersed boundary method, *Acta numerica*, *11*, 479–517.

[251] Peters, N. (1988), Laminar flamelet concepts in turbulent combustion, in *Symposium (international) on combustion*, vol. 21, pp. 1231–1250, Elsevier.

[252] Peters, N. (2001), Turbulent combustion.

[253] Peters, N., and M. Dekena (1999), Combustion modeling with the g-equation, *Oil & Gas Science and Technology*, *54*(2), 265–270.

[254] Petrarolo, A., A. Ruettgers, and M. Kobald (2019), Data clustering of hybrid rocket combustion flame, in *AIAA Propulsion and Energy 2019 Forum*, p. 4193.

[255] Pierce, C. D. (2001), *Progress-variable approach for large-eddy simulation of turbulent combustion*, stanford university.

[256] Pierce, C. D., and P. Moin (2004), Progress-variable approach for large-eddy simulation of non-premixed turbulent combustion, *Journal of fluid Mechanics*, *504*, 73–97.

[257] Pirozzoli, S. (2002), Conservative hybrid compact-weno schemes for shock-turbulence interaction, *Journal of Computational Physics*, *178*(1), 81–117.

[258] Pitsch, H. (2006), Large-eddy simulation of turbulent combustion, *Annual Review of Fluid Mechanics*, *38*(1), 453–482, doi:10.1146/annurev.fluid.38.050304.092133.

[259] Pitsch, H., O. Desjardins, G. Balarac, and M. Ihme (2008), Large-eddy simulation of turbulent reacting flows, *Progress in Aerospace Sciences*, *44*(6), 466–478.

[260] Plewa, T., T. Linde, V. G. Weirs, et al. (2005), *Adaptive mesh refinement-theory and applications*, Springer.

[261] Poinsot, T., and D. Veynante (2005), *Theoretical and numerical combustion*, RT Edwards, Inc.

[262] Polifke, W. (2014), Black-box system identification for reduced order model construction, *Annals of Nuclear Energy*, *67*, 109–128.

[263] Pope, S. (1997), Computationally efficient implementation of combustion chemistry using in situ adaptive tabulation, *Combustion Theory and Modelling*, *1*(1), 41–63, doi:10.1080/713665229.

[264] Pope, S. B. (1985), Pdf methods for turbulent reactive flows, *Progress in energy and combustion science*, *11*(2), 119–192.

[265] Pope, S. B. (2004), Ten questions concerning the large-eddy simulation of turbulent flows, *New journal of Physics*, *6*(1), 35.

[266] Pope, S. B. (2010), Self-conditioned fields for large-eddy simulations of turbulent flows, *Journal of Fluid Mechanics*, *652*, 139–169.

[267] Pope, S. B., and S. B. Pope (2000), *Turbulent flows*, Cambridge university press.

[268] Pope, S. B., and Z. Ren (2009), Efficient implementation of chemistry in computational combustion, *Flow, turbulence and combustion*, *82*(4), 437–453.

[269] Popp, S., F. Hunger, S. Hartl, D. Messig, B. Coriton, J. H. Frank, F. Fuest, and C. Hasse (2015), Les flamelet-progress variable modeling and measurements of a turbulent partially-premixed dimethyl ether jet flame, *Combustion and Flame*, *162*(8), 3016–3029.

[270] Powers, J. M., and S. Paolucci (2005), Accurate spatial resolution estimates for reactive supersonic flow with detailed chemistry, *AIAA journal*, *43*(5), 1088–1099.

[271] Prakash, S., and V. Raman (2021), The effects of mixture preburning on detonation wave propagation, *Proceedings of the Combustion Institute*, *38*(3), 3749–3758.

[272] Prakash, S., R. Fiévet, and V. Raman (2019), The effect of fuel stratification on the detonation wave structure, in *AIAA Scitech 2019 Forum*, p. 1511.

[273] Prakash, S., R. Fiévet, V. Raman, J. Burr, and K. H. Yu (2020), Analysis of the detonation wave structure in a linearized rotating detonation engine, *AIAA Journal*, *58*(12), 5063–5077.

[274] Prakash, S., V. Raman, C. F. Lietz, W. A. Hargus Jr, and S. A. Schumaker (2021), Numerical simulation of a methane-oxygen rotating detonation rocket engine, *Proceedings of the Combustion Institute*, *38*(3), 3777–3786.

[275] Raissi, M., P. Perdikaris, and G. E. Karniadakis (2019), Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational physics*, *378*, 686–707.

[276] Raman, V., and M. Hassanaly (2019), Emerging trends in numerical simulations of combustion systems, *Proceedings of the Combustion Institute*, *37*(2), 2073–2089.

[277] Raman, V., and H. Pitsch (2007), A consistent les/filtered-density function formulation for the simulation of turbulent flames with detailed chemistry, *Proceedings of the Combustion Institute*, *31*(2), 1711–1719.

[278] Raman, V., S. Prakash, and M. Gamba (2022), Non-idealities in rotating detonation engines, *Annual Review of Fluid Mechanics*, *48*, 159–190.

[279] Ramezanian, D., A. G. Nouri, and H. Babaee (2021), On-the-fly reduced order modeling of passive and reactive species via time-dependent manifolds, *Computer Methods in Applied Mechanics and Engineering*, *382*, 113,882.

[280] Ranade, R., and T. Echekki (2019), A framework for data-based turbulent combustion closure: A posteriori validation, *Combustion and flame*, *210*, 279–291.

[281] Rankin, B. A., M. L. Fotia, A. G. Naples, C. A. Stevens, J. L. Hoke, T. A. Kaemming, S. W. Theuerkauf, and F. R. Schauer (2017), Overview of performance, application, and analysis of rotating detonation engine technologies, *Journal of Propulsion and Power*, *33*(1), 131–143.

[282] Rankin, B. A., D. R. Richardson, A. W. Caswell, A. G. Naples, J. L. Hoke, and F. R. Schauer (2017), Chemiluminescence imaging of an optically accessible non-premixed rotating detonation engine, *Combustion and Flame*, *176*, 12–22.

[283] Roache, P. J. (2002), Code verification by the method of manufactured solutions, *J. Fluids Eng.*, *124*(1), 4–10.

[284] Robinson, C. (1998), *Dynamical systems: stability, symbolic dynamics, and chaos*, CRC press.

[285] Rousseeuw, P. J. (1987), Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, *Journal of computational and applied mathematics*, *20*, 53–65.

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

[287] Rowley, C. W., I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson (2009), Spectral analysis of nonlinear flows, *Journal of fluid mechanics*, *641*, 115–127.

[288] Rubner, Y., C. Tomasi, and L. J. Guibas (2000), The earth mover's distance as a metric for image retrieval, *International journal of computer vision*, *40*(2), 99–121.

[289] Rudin, C., C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong (2022), Interpretable machine learning: Fundamental principles and 10 grand challenges, *Statistics Surveys*, *16*, 1–85.

[290] Saghafian, A., L. Shunn, D. A. Philips, and F. Ham (2015), Large eddy simulations of the hifire scramjet using a compressible flamelet/progress variable approach, *Proceedings of the Combustion Institute*, *35*(2), 2163–2172.

[291] Saghafian, A., L. Shunn, D. A. Philips, and F. Ham (2015), Large eddy simulations of the hifire scramjet using a compressible flamelet/progress variable approach, *Proceedings of the Combustion Institute*, *35*(2), 2163–2172.

[292] Saghafian, A., V. E. Terrapon, and H. Pitsch (2015), An efficient flamelet-based combustion model for compressible flows, *Combustion and Flame*, *162*(3), 652–667.

[293] San, O., and T. Iliescu (2013), Proper orthogonal decomposition closure models for fluid flows: Burgers equation, *arXiv preprint arXiv:1308.3276*.

[294] Sargent, R. G. (2010), Verification and validation of simulation models, in *Proceedings of the 2010 winter simulation conference*, pp. 166–183, IEEE.

[295] Sato, T. (2020), High-fidelity simulations for rotating detonation engines, Ph.D. thesis, University of Michigan, Ann Arbor.

[296] Sato, T., S. Voelkel, and V. Raman (2018), Detailed chemical kinetics based simulation of detonation-containing flows, in *Turbo Expo: Power for Land, Sea, and Air*, vol. 51050, p. V04AT04A063, American Society of Mechanical Engineers.

[297] Sato, T., F. Chacon, M. Gamba, and V. Raman (2021), Mass flow rate effect on a rotating detonation combustor with an axial air injection, *Shock Waves*, *31*(7), 741–751.

[298] Sato, T., F. Chacon, L. White, V. Raman, and M. Gamba (2021), Mixing and detonation structure in a rotating detonation engine with an axial air inlet, *Proceedings of the Combustion Institute*, *38*(3), 3769–3776.

[299] Saxena, A., M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, and C.-T. Lin (2017), A review of clustering techniques and developments, *Neurocomputing*, *267*, 664–681.

[300] Schmid, P. J. (2010), Dynamic mode decomposition of numerical and experimental data, *Journal of fluid mechanics*, *656*, 5–28.

[301] Schmid, P. J. (2022), Dynamic mode decomposition and its variants, *Annual Review of Fluid Mechanics*, *54*, 225–254.

[302] Schneider, D. (2022), The exascale era is upon us: The frontier supercomputer may be the first to reach 1,000,000,000,000,000,000 operations per second, *IEEE Spectrum*, *59*(1), 34–35.

[303] Schulze, P., J. Reiss, and V. Mehrmann (2019), Model reduction for a pulsed detonation combuster via shifted proper orthogonal decomposition, in *Active Flow and Combustion Control 2018*, pp. 271–286, Springer.

[304] Schwer, D., and K. Kailasanath (2011), Numerical investigation of the physics of rotating-detonation-engines, *Proceedings of the combustion institute*, *33*(2), 2195–2202.

[305] Seiner, J. M., S. Dash, and D. Kenzakowski (2001), Historical survey on enhanced mixing in scramjet engines, *Journal of Propulsion and Power*, *17*(6), 1273–1286.

[306] Sen, B. A., and S. Menon (2009), Turbulent premixed flame modeling using artificial neural networks based chemical kinetics, *Proceedings of the Combustion Institute*, *32*(1), 1605–1611.

[307] Sen, B. A., and S. Menon (2010), Linear eddy mixing based tabulation and artificial neural networks for large eddy simulations of turbulent flames, *Combustion and Flame*, *157*(1), 62–74.

[308] Sewerin, F., and S. Rigopoulos (2015), A methodology for the integration of stiff chemical kinetics on gpus, *Combustion and Flame*, *162*(4), 1375–1394.

[309] Shalf, J. (2020), The future of computing beyond moore's law, *Philosophical Transactions of the Royal Society A*, *378*(2166), 20190,061.

[310] Sharma, A. J., R. F. Johnson, D. A. Kessler, and A. Moses (2020), Deep learning for scalable chemical kinetics, in *AIAA scitech 2020 forum*, p. 0181.

[311] Shepherd, J. E. (2009), Detonation in gases, *Proceedings of the Combustion Institute*, *32*(1), 83–98.

[312] Shepherd, J. E. (2021), Caltech Shock and Detonation Toolbox, *Explosion Dynamics Laboratory.*

[313] Short, M., and J. J. Quirk (1997), On the nonlinear stability and detonability limit of a detonation wave for a model three-step chain-branching reaction, *Journal of Fluid Mechanics*, *339*, 89–119.

[314] Shunn, L., F. Ham, and P. Moin (2012), Verification of variable-density flow solvers using manufactured solutions, *Journal of Computational Physics*, *231*(9), 3801–3827.

[315] Sirovich, L. (1987), Turbulence and the dynamics of coherent structures. i. coherent structures, *Quarterly of applied mathematics*, *45*(3), 561–571.

[316] Sirovich, L., B. Knight, and J. Rodriguez (1990), Optimal low-dimensional dynamical approximations, *Quarterly of applied mathematics*, *48*(3), 535–548.

[317] Slotnick, J. P., A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. J. Mavriplis (2014), Cfd vision 2030 study: a path to revolutionary computational aerosciences.

[318] Smagorinsky, J. (1963), General circulation experiments with the primitive equations: I. the basic experiment, *Monthly weather review*, *91*(3), 99–164.

[319] Smith, G., Y. Tao, and H. Wang (2016), Foundational fuel chemistry model version 1.0 (FFCM-1), `http://nanoenergy.stanford.edu/ffcm1`.

[320] Soize, C., and R. Ghanem (2022), Probabilistic learning on manifolds (plom) with partition, *International Journal for Numerical Methods in Engineering*, *123*(1), 268–290.

[321] Sportisse, B. (2000), An analysis of operator splitting techniques in the stiff case, *Journal of computational physics*, *161*(1), 140–168.

[322] Steinberg, A. M., I. Boxx, M. Stöhr, C. D. Carter, and W. Meier (2010), Flow–flame interactions causing acoustically coupled heat release fluctuations in a thermo-acoustically unstable gas turbine model combustor, *Combustion and Flame*, *157*(12), 2250–2266.

[323] Steinley, D. (2006), K-means clustering: a half-century synthesis, *British Journal of Mathematical and Statistical Psychology*, *59*(1), 1–34.

[324] Stöhr, M., I. Boxx, C. Carter, and W. Meier (2011), Dynamics of lean blowout of a swirl-stabilized flame in a gas turbine model combustor, *Proceedings of the Combustion Institute*, *33*(2), 2953–2960.

[325] Strang, G. (1968), On the construction and comparison of difference schemes, *SIAM journal on numerical analysis*, *5*(3), 506–517.

[326] Su, L. K., and W. J. Dahm (1996), Scalar imaging velocimetry measurements of the velocity gradient tensor field in turbulent flows. ii. experimental results, *Physics of Fluids*, *8*(7), 1883–1906.

[327] Suchocki, J., S.-T. Yu, J. Hoke, A. Naples, F. Schauer, and R. Russo (2012), Rotating detonation engine operation, in *50th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*, p. 119.

[328] Sun, L., H. Gao, S. Pan, and J.-X. Wang (2020), Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, *Computer Methods in Applied Mechanics and Engineering*, *361*, 112,732.

[329] Taamallah, S., Z. A. LaBry, S. J. Shanbhogue, and A. F. Ghoniem (2015), Thermo-acoustic instabilities in lean premixed swirl-stabilized combustion and their link to acoustically coupled and decoupled flame macrostructures, *Proceedings of the combustion institute*, *35*(3), 3273–3282.

[330] Taira, K., et al. (2017), Modal analysis of fluid flows: An overview, *Aiaa Journal*, *55*(12), 4013–4041.

[331] Tang, Y., and V. Raman (2021), Large eddy simulation of premixed turbulent combustion using a non-adiabatic, strain-sensitive flamelet approach, *Combustion and Flame*, *234*, 111,655.

[332] Thumuluru, S. K., and T. Lieuwen (2009), Characterization of acoustically forced swirl flame dynamics, *Proceedings of the Combustion Institute*, *32*(2), 2893–2900.

[333] Titarev, V. A., and E. F. Toro (2004), Finite-volume weno schemes for three-dimensional conservation laws, *Journal of Computational Physics*, *201*(1), 238–260.

[334] Titi, E. S. (1990), On approximate inertial manifolds to the navier-stokes equations, *Journal of mathematical analysis and applications*, *149*(2), 540–557.

[335] Toro, E. F. (2013), *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*, Springer Science & Business Media.

[336] Towery, C. A., K. M. Smith, P. Shrestha, P. E. Hamlington, and M. Van Schoor (2014), Examination of turbulent flow effects in rotating detonation engines, in *44th AIAA Fluid Dynamics Conference*, p. 3031.

[337] Towne, A., O. T. Schmidt, and T. Colonius (2018), Spectral proper orthogonal decomposition and its relationship to dynamic mode decomposition and resolvent analysis, *Journal of Fluid Mechanics*, *847*, 821–867.

[338] Tseng, G. C. (2007), Penalized and weighted k-means for clustering with scattered objects and prior information in high-throughput biological data, *Bioinformatics*, *23*(17), 2247–2255.

[339] Tu, J. H. (2013), Dynamic mode decomposition: Theory and applications, Ph.D. thesis, Princeton University.

[340] Uranakara, H., S. Barwey, F. Pérez, V. Vijayaranga, V. Raman, and H. Im (2022), Accelerating turbulent reacting flow simulations on many-core/GPUs using matrix-based kinetics, *Accepted to Proceedings of the Combustion Institute*.

[341] Valorani, M., D. A. Goussis, F. Creta, and H. N. Najm (2005), Higher order corrections in the approximation of low-dimensional manifolds and the construction of simplified problems with the csp method, *Journal of Computational Physics*, *209*(2), 754–786.

[342] Van Leer, B. (1979), Towards the ultimate conservative difference scheme. v. a second-order sequel to godunov's method, *Journal of computational Physics*, *32*(1), 101–136.

[343] Van Oijen, J., A. Donini, R. Bastiaans, J. ten Thije Boonkkamp, and L. De Goey (2016), State-of-the-art in premixed combustion modeling using flamelet generated manifolds, *Progress in Energy and Combustion Science*, *57*, 30–74.

[344] Verfürth, R. (1994), A posteriori error estimation and adaptive mesh-refinement techniques, *Journal of Computational and Applied Mathematics*, *50*(1-3), 67–83.

[345] Von Luxburg, U. (2007), A tutorial on spectral clustering, *Statistics and computing*, *17*(4), 395–416.

[346] Walker, D. W., and J. J. Dongarra (1996), MPI: a standard message passing interface, *Supercomputer*, *12*, 56–68.

[347] Wall, C., B. J. Boersma, and P. Moin (2000), An evaluation of the assumed beta probability density function subgrid-scale model for large eddy simulation of nonpremixed, turbulent combustion with heat release, *Physics of fluids*, *12*(10), 2522–2529.

[348] Wan, K., C. Barnaud, L. Vervisch, and P. Domingo (2020), Chemistry reduction using machine learning trained from non-premixed micro-mixing modeling: Application to dns of a syngas turbulent oxy-flame with side-wall effects, *Combustion and Flame*, *220*, 119–129.

[349] Wang, Q., M. Ihme, Y.-F. Chen, and J. Anderson (2022), A tensorflow simulation framework for scientific computing of fluid flows on tensor processing units, *Computer Physics Communications*, *274*, 108,292.

[350] Wei, Z., Z. Yang, C. Xia, and Q. Li (2017), Cluster-based reduced-order modelling of the wake stabilization mechanism behind a twisted cylinder, *Journal of Wind Engineering and Industrial Aerodynamics*, *171*, 288–303.

[351] Wen, X., K. Luo, Y. Luo, H. Wang, and J. Fan (2018), Large-eddy simulation of multiphase combustion jet in cross-flow using flamelet model, *International Journal of Multiphase Flow*, *108*, 211–225.

[352] Wienke, S., P. Springer, C. Terboven, et al. (2012), OpenACC—first experiences with real-world applications, in *European Conference on Parallel Processing*, pp. 859–870, Springer.

[353] Willcox, K. (2006), Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition, *Computers & fluids*, *35*(2), 208–226.

[354] Williams, S., A. Waterman, and D. Patterson (2009), Roofline: an insightful visual performance model for multicore architectures, *Communications of the ACM*, *52*(4), 65–76.

[355] Wolanski, P., J. Kindracki, T. Fujiwara, Y. Oka, and K. Shima-uchi (2005), An experimental study of rotating detonation engine, in *20th International Colloquium on the Dynamics of Explosions and Reactive Systems*, vol. 31.

[356] Wold, S., K. Esbensen, and P. Geladi (1987), Principal component analysis, *Chemometrics and intelligent laboratory systems*, *2*(1-3), 37–52.

[357] Wu, H., P. C. Ma, and M. Ihme (2019), Efficient time-stepping techniques for simulating turbulent reactive flows with stiff chemistry, *Computer Physics Communications*, *243*, 81–96.

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

[359] Xu, R., and H. Wang (2018), Reduced reaction models for methane and ethylene combustion, personal communication.

[360] Xu, R., et al. (2018), A physics-based approach to modeling real-fuel combustion chemistry–ii. reaction kinetic models of jet and rocket fuels, *Combustion and Flame*, *193*, 520–537.

[361] Xu, R., et al. (2018), A physics-based approach to modeling real-fuel combustion chemistry–ii. reaction kinetic models of jet and rocket fuels, *Combustion and Flame*, *193*, 520–537.

[362] Yang, B., and S. Pope (1998), An investigation of the accuracy of manifold methods and splitting schemes in the computational implementation of combustion chemistry, *Combustion and Flame*, *112*(1-2), 16–32.

[363] Yao, W., Y. Yuan, X. Li, J. Wang, K. Wu, and X. Fan (2018), Comparative study of elliptic and round scramjet combustors fueled by rp-3, *Journal of Propulsion and Power*, *34*(3), 772–786.

[364] Yungster, S., and K. Radhakrishnan (2004), Pulsating one-dimensional detonations in hydrogen–air mixtures, *Combustion Theory and Modelling*, *8*(4), 745.

[365] Zagaris, A., H. G. Kaper, and T. J. Kaper (2004), Analysis of the computational singular perturbation reduction method for chemical kinetics, *Journal of Nonlinear Science*, *14*(1), 59–91.

[366] Zhang, J., G. Wu, X. Hu, S. Li, and S. Hao (2011), A parallel k-means clustering algorithm with mpi, in *2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming*, pp. 60–64, IEEE.

[367] Zhang, W., et al. (2019), Amrex: a framework for block-structured adaptive mesh refinement, *Journal of Open Source Software*, *4*(37), 1370–1370.

[368] Zhang, Y., L. Zhou, H. Meng, and H. Teng (2020), Reconstructing cellular surface of gaseous detonation based on artificial neural network and proper orthogonal decomposition, *Combustion and Flame*, *212*, 156–164.

[369] Zhou, L., Y. Song, W. Ji, and H. Wei (2022), Machine learning for combustion, *Energy and AI*, *7*, 100,128.